AUTOMATIC CLASSIFICATION OF VOLCANIC EARTHQUAKES USING MULTI-STATION WAVEFORMS AND DYNAMIC NEURAL NETWORKS

By

Christopher Patrick Bruton

RECOMMENDED:

Carl A. Tape

Dr. Carl Tape
Advisory Committee Member

Dr. Jeffrey Freymueller
Advisory Committee Member

Dr. Michael West
Advisory Committee Chair

Dr. Paul McCarthy
Chair, Department of Geology and Geophysics

APPROVED:

Dr. Paul Layer
Dean, College of Natural Science and Mathematics

Dr. John Eichelberger
Dean of the Graduate School

4/21/14

Date

AUTOMATIC CLASSIFICATION OF VOLCANIC EARTHQUAKES USING MULTI-STATION WAVEFORMS AND DYNAMIC NEURAL NETWORKS

A

THESIS

Presented to the Faculty

of the University of Alaska Fairbanks

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

By

Christopher Patrick Bruton, B.Sc.

Fairbanks, Alaska

May 2014

Abstract

Earthquakes and seismicity have long been used to monitor volcanoes. In addition to the time, location, and magnitude of an earthquake, the characteristics of the waveform itself are important. For example, low-frequency or hybrid type events could be generated by magma rising toward the surface. A rockfall event could indicate a growing lava dome. Classification of earthquake waveforms is thus a useful tool in volcano monitoring. A procedure to perform such classification automatically could flag certain event types immediately, instead of waiting for a human analyst's review.

Inspired by speech recognition techniques, we have developed a procedure to classify earthquake waveforms using artificial neural networks. A neural network can be "trained" with an existing set of input and desired output data; in this case, we use a set of earthquake waveforms (input) that has been classified by a human analyst (desired output). After training the neural network, new sets of waveforms can be classified automatically as they are presented.

Our procedure uses waveforms from multiple stations, making it robust to seismic network changes and outages. The use of a dynamic time-delay neural network allows waveforms to be presented without precise alignment in time, and thus could be applied to continuous data or to seismic events without clear start and end times. We have evaluated several different training algorithms and neural network structures to determine their effects on classification performance.

We apply this procedure to earthquakes recorded at Mount Spurr and Katmai in Alaska, and Uturuncu Volcano in Bolivia. The procedure can successfully distinguish between slab and volcanic events at Uturuncu, between events from four different volcanoes in the Katmai region, and between volcano-tectonic and long-period events at Spurr. Average recall and overall accuracy were greater than 80% in all three cases.

Table of Contents

Page

List of Figures

List of Tables

Page

Acknowledgements

I would like to acknowledge my advisor, Michael West, for inviting me to Alaska, for providing the initial inspiration for this project, for countless ideas and suggestions, and for pushing me along when motivation was needed. I would also like to thank my other committee members, Jeffrey Freymueller and Carl Tape, as well as my former committee members Stephen McNutt and Glenn Thompson, for their valuable input on this work.

In addition to my committee members, I received input and support from my fellow volcanology and seismology students during my time working on this project. Many people provided feedback, but in particular, I would like to acknowledge Helena Buurman and Ophelia George.

I would also like to thank my colleagues at the Geophysical Institute, Alaska Volcano Observatory, and Alaska Earthquake Center for providing the resources and support necessary to complete this project. Financial support was provided by the Alaska Volcano Observatory and the National Science Foundation (PLUTONS project).

I would like to thank my family in Calgary, Alberta, Canada for supporting me from afar: my parents Don and Patty Bruton, and my siblings Rebecca and Alex.

Finally I would like to thank the many friends who have made Fairbanks a fun place to be over the past four years. I couldn't have done it without you.

—Christopher Bruton, February 2014

Évidemment sur notre terre nous sommes beaucoup trop petits pour ramoner nos volcans. C'est pourquoi ils nous causent des tas d'ennuis.

—Antoine de Saint-Exupéry

Chapter 1
Introduction

Earthquakes and seismicity have long been used to monitor volcanoes. In addition to the time, location, and magnitude of an earthquake, the characteristics of the waveforms themselves are important. For example, low-frequency or hybrid type events are generated by magma rising toward the surface. A rockfall event can indicate a growing lava dome. Classification of earthquake waveforms is thus a useful tool in volcano monitoring. In this thesis, we[1] present a new method to automatically classify earthquake waveforms using artificial neural networks.

This chapter details the basic motivation and background for this project. Chapter 2 describes the classification procedure that we have developed, Chapter 3 describes three case studies to which we apply the procedure, and Chapter 5 contains our overall conclusions. The Appendix provides some additional detail about the methods described in Chapter 2.

1.1 Motivation

There are several different motivations for a procedure to automatically classify earthquakes from their waveforms.

First, it is important as an analysis tool for large data sets. It is time-consuming to manually classify a large amount of earthquakes. Using a generous estimate of 5 seconds per human classification decision, a set of 10000 events would take 14 hours to get through. 100000 events would take over 17 8-hour work days to manually classify. This is not insurmountably arduous, but we suspect most seismologists would prefer to spend their time otherwise.

A human-trained automatic classification procedure could reduce the manual classification requirement to a small subset of the total earthquakes. For example, a random sample of 1000 earthquakes could be manually classified, then used to train the automatic classifier. If the system is well-designed, the classifier will mimic the human's choices and classify the remaining 99000 earthquakes within seconds.

There is also a greater degree of consistency in such a system. The quality of manual classification of 100000 earthquakes could vary with the seismologist's mood and the

[1] In this thesis, 'we', 'our', 'us', etc. refer to the author, Christopher Bruton, and his graduate advisor Michael West. West will be listed as co-author on any future publications resulting from this thesis.

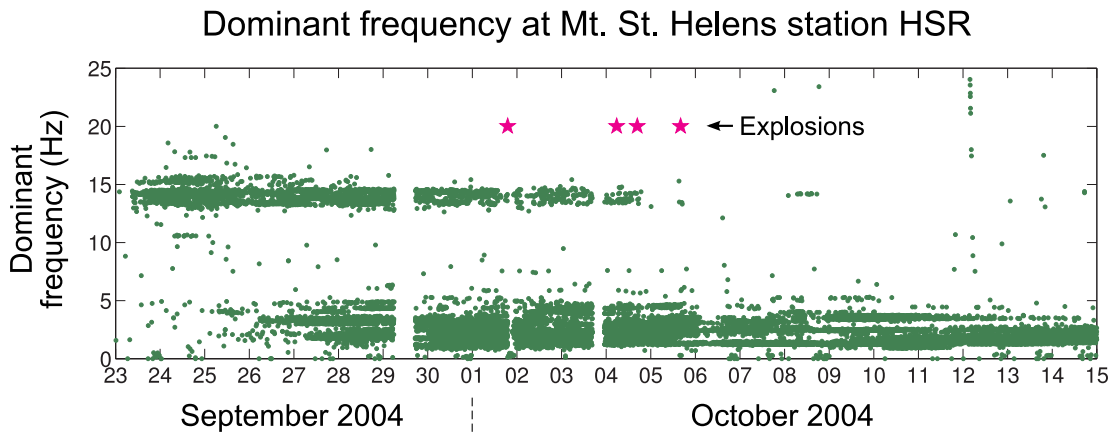

Figure 1.1. Frequency transition before explosions at Mount St. Helens. A transition from high-frequency to low-frequency earthquakes corresponded to phreatic explosions during the 2004 eruption of Mount St. Helens.

closeness of his or her deadline. There is even less possibility of consistency if the work is divided between multiple analysts. With a trained system, a single seismologist could classify 1000 earthquakes in one afternoon, and be confident that the remaining 99000 classifications be consistent with his or her own.

Why do seismologists want to classify earthquakes in the first place? In volcano seismology, different types of earthquakes can mean different things about what a volcano is doing. For example, low-frequency or hybrid type events are associated with fluid movement through a conduit, possibly indicating magma rising toward the surface. A rockfall event, caused by landslides on the flanks of a volcano as it inflates, often indicates a growing lava dome. Figure 1.1 shows how a transition from high-frequency to low-frequency earthquakes corresponded to phreatic explosions during the 2004 eruption of Mount St. Helens—a direct example of how seismic event types can relate to volcanic activity [*Moran et al.*, 2008].

Naturally, automatic classification has applications to real-time monitoring of volcanoes and eruption warning systems. If a volcano has a precedent for producing certain types of earthquakes prior to an eruption, an automatic classification system can flag these earthquakes as they occur and immediately alert a human analyst. Without an automatic system, it could be hours or days after an event before an analyst looks at its waveforms. An automatic classification system could also flag earthquakes that it *does not recognize*—

and alert an analyst that it may not be business as usual at the volcano.

Automatic classification also has applications outside of volcano monitoring. In any branch of seismology, human-generated noise, instrument calibration pulses, or other undesirable signals are usually unwanted in the data. An automatic classification procedure could screen out these signals as they occur. In the field of earthquake early warning, harmless volcanic earthquakes must be immediately screened out. Their waveforms may appear very similar to the body-wave precursors of damaging tectonic events, so they must be ignored in order to avoid false alarms. Volcanic seismicity and monitoring are the focus of this particular project, but we emphasize that the uses of automatic classification extend beyond volcanoes.

1.2 Background

Automatic classification of earthquakes is not a new field of study. *Falsaperla et al.* [1996] had modest success with a neural network-based classification system for explosion earthquakes at Mount Stromboli. Over the past two decades, computer processing power and storage capacity has increased greatly. Classification capability applied to fields such as speech processing and recognition has improved in turn. But seismic event classification has not seen the same kinds of advancements—perhaps due to the commercial value of adept speech recognition software, and lack of such value for seismological applications.

Yıldırım et al. [2011] tested three different algorithms to distinguish between earthquakes and quarry blasts; while they obtained very good performance (97.67%–100%), their data set comprised only 175 events, and their classifier input comprised two scalar parameters: the S/P wave amplitude ratio, and "complexity", derived from the integrated velocity seismogram. Reliance on pre-defined scalar parameters limits the usefulness of a technique, because such parameters do not generalize to new classification problems.

Other groups have made stronger advances in the field. *Beyreuther and Wassermann* [2008] successfully used a Hidden Markov Model approach to identify and classify earthquakes within continuous waveform data. However, we have not yet seen a study that used seismograms from multiple stations, *at the same time*, to automatically classify earthquakes. Seismologists have been using large seismic networks for decades to monitor and locate earthquakes: if an event is recorded at ten stations, why should we ignore nine of those seismograms when performing automatic classification?

The goal of this project was to develop a procedure that can use seismograms from

many stations to perform automatic classification. Using more of the available data ought to provide a performance improvement over single-station methods. It also makes the classification procedure more robust to data outages and changing seismic network configurations, because there is no continual reliance on data from any one station. This is key for any real-time monitoring system.

As mentioned above, published automatic classification studies have tended to use small data sets with straightforward earthquake classes. We take this much further in this project, with larger data sets and varied classification problems, and show very good classification performance using artificial neural networks.

Chapter 2
Methods

Our classification procedure is a multi-step procedure that requires preparation of datasets, pre-processing, neural network training, neural network simulation, and post-processing. This chapter describes the classification procedure in a general way. Specific datasets and results are presented in Chapter 3. To reduce ambiguity we refer to our procedure in this chapter as the WC2 procedure[1].

2.1 Defining a Classification Problem

In order to clarify exactly what the WC2 procedure aims to accomplish, let us set aside the details for a moment and regard it as a black box. What kinds of problems can it solve and what sort of input is required?

The WC2 procedure is a trained classifier. It is designed to learn to classify waveforms based on existing, pre-classified data sets. For example, suppose we record 1000 earthquakes and classify each of them as either a volcanic event or a tectonic event. We could then provide this information (both the waveforms and the classifications we have made) to the WC2 black box during the training process. The black box would then be ready to attempt to classify any new waveform presented to it. Suppose the 1000 training earthquakes were drawn from a random sample of 100000 recorded earthquakes: after training the black box, we might use it to automatically classify the remaining 99000 earthquakes.

The particular classification scheme used is entirely dependent on the problem to solve. In our case studies (Chapter 3) we have one location-based classification scheme and two based on volcanic earthquake types (LP, VT, etc.). During the course of this project, however, we received other suggestions for classification problems, including identifying instrument calibration pulses, identifying high-quality SKS phases for mantle anisotropy studies, and earthquake early warning systems. The exact criteria for classifying training events is therefore dependent on the classification scheme. In many cases, classification will be based on a visual inspection of the waveforms or their spectrograms. But event types could also be assigned based on location, depth, or other criteria.

The WC2 black box makes use of the following input: 1. A set of waveforms to use for training; 2. A predefined classification (i.e. training target) associated with each training

[1] Waveform Classification II. Waveform Classification I (WC1) was a rudimentary early version of this procedure.

event; and 3. A set of new or unclassified waveforms to present after the black box has been trained. The details of how to prepare these input data will be discussed in subsequent sections of this chapter.

A key requirement for any automatic classification procedure is that the information required to make a classification decision is actually present in the input data. For the WC2 procedure, this means that a classification must be possible based on characteristics of the waveforms themselves, despite the fact that an experienced human analyst preparing a training data set may use additional information to make his or her classifications. For example, suppose the goal is to distinguish between volcanic and tectonic events; if the analyst knows the actual location of each event, he or she might use these locations to help choose each event type. These locations will not be available to the WC2 procedure, so automatic classification will only be successful if waveform characteristics of the volcanic and tectonic events also differ in some way[2].

2.2 Time-delay Neural Networks

We now open the black box and take a look inside. At the core of the WC2 classification procedure is an artificial neural network (ANN, NN, or neural network for short). A neural network is a collection of interconnected nodes, or neurons. Each neuron performs a calculation on one or more input values, producing a single output value. Output values may then be passed on to subsequent neurons, or passed as final output from the neural network. The calculation performed by each neuron is a combination of a weighted sum and an activation function (Figure 2.1). With the right network structure and combination of neuron weights, a neural network can perform complex nonlinear calculations and tasks such as classification [*Masters*, 1993].

The key to neural networks' power lies in how the weights are determined. They are not explicitly chosen by a human operator, but are rather calculated by an iterative training or learning procedure such that they minimize some measure of performance or error. The WC2 procedure uses a supervised network, meaning that it attempts to learn to make decisions based on target outputs provided by the operator. In this case, the operator provides some already-classified earthquake data to the network, and the network learns to mimic these classifications during the training process. This is in contrast to an unsuper-

[2] A corollary to this requirement is that successful automatic classification based on waveforms alone *proves* that distinguishing characteristics exist within the waveforms, even if they are not immediately apparent to human analysts.

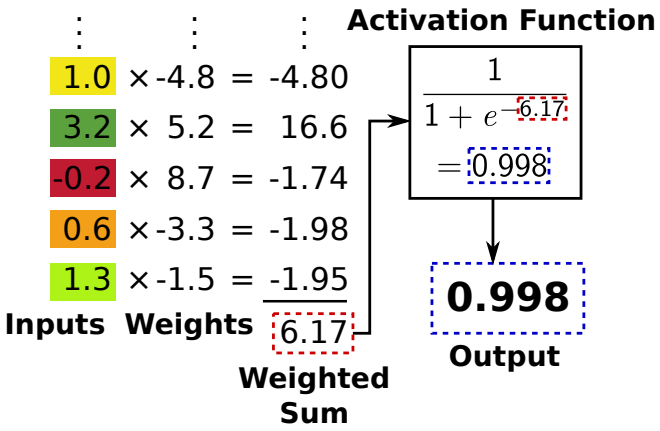

Figure 2.1. Inside an individual neuron. Each input value is first multiplied by its associated weight. These values are then summed and passed through an activation function to determine the final output value. The weights are determined during training and remain unchanged thereafter. Here, only five input-weight pairs are shown, but there will actually be one weight value per input position (i.e. per spectrogram value within the sliding window). The activation function shown here is just an example—most any sigmoid function is possible to use (Section A.6). Figure 2.4 shows how multiple neurons fit together to form a complete neural network.

vised network, which would separate the data and establish waveform patterns on its own [*Masters*, 1993]. One particular style of unsupervised network is a Self-Organizing Map; this has also been used in seismology for earthquake classification [e.g. *Köhler et al.*, 2010], but we find a supervised network more appropriate for our goals.

The WC2 procedure uses a time-delay neural network, a type of dynamic neural network introduced by *Waibel et al.* [1989] to classify phonemes in human speech. Dynamic neural networks differ from static neural networks mainly in how sequential data (typically time series, such as waveforms or spectrograms) are presented and interpreted. With a static neural network, time series are presented in separate and distinct chunks, and a single set of output values is calculated from each chunk of input. For example, a static neural network trained to classify earthquakes might be presented with one 30-second spectrogram per earthquake, and would determine a single classification per spectrogram. A dynamic neural network is also presented with distinct chunks of data, but the chunks overlap with each other. The same 30-second spectrogram might be interpreted in 10-second chunks, one second at a time (i.e. the first chunk would contain seconds 1 through 10, the second chunk would contain seconds 2 through 11, the third would contain seconds 3 through 12, and so on). Each time point of data would be presented to the neural network up to 10 times, but in a different position each time. Furthermore, each 10-second chunk would produce its own classification output values.

In the context of earthquake classification, the main advantage of a dynamic neural network is that it is trained to make classifications regardless of the time alignment of each waveform. Whether an event occurs two seconds or 10 seconds after the start of the input spectrogram, a successfully-trained network will be able to classify it. In contrast, a static neural network requires all events to occur at exactly the same time within their input spectrograms. While this is practical to achieve in some cases, it is not suitable for all continuous applications or cases where arrival times are not precisely known.

Waibel et al. [1989] used a specialized time-delay neural network to classify phonemes based on spectrograms of recorded human speech. They obtained an average of 98.5% correct classifications on their testing data. Pragmatically, speech recognition and earthquake waveform classification are similar problems: both act on digitally sampled waveforms with rich frequency content. We decided to adopt *Waibel et al.*'s approach of presenting spectrograms to a time-delay neural network. The WC2 procedure uses a simplified version of their neural network, with delays only at the input layer and without their special

method of updating weights during training.

With a time-delay neural network at the core, the remaining steps of the WC2 procedure are essentially data management: preparing and importing data, generating spectrograms from waveforms, arranging spectrograms to present to the neural network, and processing output from the network. These steps are described in the remainder of this chapter.

2.3 Preparing a Dataset

Our starting point for classification is an earthquake catalog. The WC2 procedure is designed for events with arrivals at multiple stations. Waveform segments are extracted and aligned based on arrival times at individual stations. Input waveforms are ultimately arranged in a two-dimensional array: each row contains waveforms associated with a particular event, aligned on arrival time at each station. Each column corresponds to one channel of one station. Within this project we only use the vertical channel at each station, but it is possible to use other channels as well.

All input to the WC2 procedure is presented in this manner. For training the neural network classifier, however, additional information is required: each event must be assigned a type according to an established classification scheme (Section 2.1). Note that one type is assigned per event; types are not assigned to each individual waveform.

The training data must also be split into a training set and a testing set: the testing set is not used to train the neural network, but is used to test its performance. When the testing set is presented to the neural network, the actual network output is compared with the target output (i.e. the correct or expected event types) to check that the network can successfully classify new data and has not merely memorized features of the training set. There is no hard rule for the size of the testing set; ideally it should be as small as possible (in order to allow as many events as possible to train the neural network) while maintaining adequate variety to provide a valid measure of performance. A good starting point is to allow 25% of the data to be used for testing; this can be adjusted downward for large data sets and should be adjusted upward for very small data sets.

We have developed routines to import event catalogs and waveforms from Antelope databases into MATLAB in the format described above. These are now publicly available as part of the GISMO Waveform Suite [*Reyes and West*, 2011].

2.4 Spectrograms and Targets

After earthquake databases are prepared and waveforms imported, the next step is to generate spectrograms. Spectrograms reduce the size of the data vectors presented to the neural network without discarding the most useful information, and may highlight frequency-based features that are often used to classify seismic events [e.g. *De Angelis and McNutt*, 2007]. Similar approaches are used in analogous problems in speech recognition [*Waibel et al.*, 1989].

It is possible to use waveforms themselves instead of spectrograms, but there are several disadvantages to this approach. First, the volume of input data would vastly increase. For example, 5 seconds of seismic data sampled at 100 Hz would contain 500 separate values, whereas a spectrogram might only contain 50 values (depending on chosen frequency bands and window sizes). Each input value has a corresponding weight within each neuron in the first layer of the neural network, and as more weights are added to a network, more data is needed to adequately train it [*Masters*, 1993]. Second, converting to spectrograms is a way to ensure that data dimensions are consistent across stations. If a seismic network contains some 50 Hz channels and some 100 Hz channels, it is problematic to present waveforms from both types of digitizer to the neural network simultaneously. Converting to spectrograms with one-second time steps allows the data from all stations to be presented at once. Finally, neural networks are powerful but not magical: it makes sense to simplify the input data and, wherever possible, highlight features that are known to be important. When a human analyst classifies a waveform, he or she is not looking at every individual sample, but rather how periods and amplitudes change over time. A spectrogram is an ideal way to represent and highlight this information.

In order to generate spectrograms, their properties must be chosen: windowing function, window length, overlap length, and fast Fourier transform (FFT) length. We allow these lengths to be specified by time, rather than number of samples, so that spectrogram time steps will be consistent between waveforms with different sample rates (this means that some spectrograms may have different numbers of frequency bands than others, but that is fine because of how they are arranged; see Figure 2.3). The spectrogram properties should be chosen such that they emphasize features distinguishing between event types. For example, our datasets contain mainly short (5–15 s) nearby events, so we chose to use one-second windows in our spectrograms. This provides adequate frequency resolution in the range (around 2–10 Hz) of interest for these events. If we were instead studying

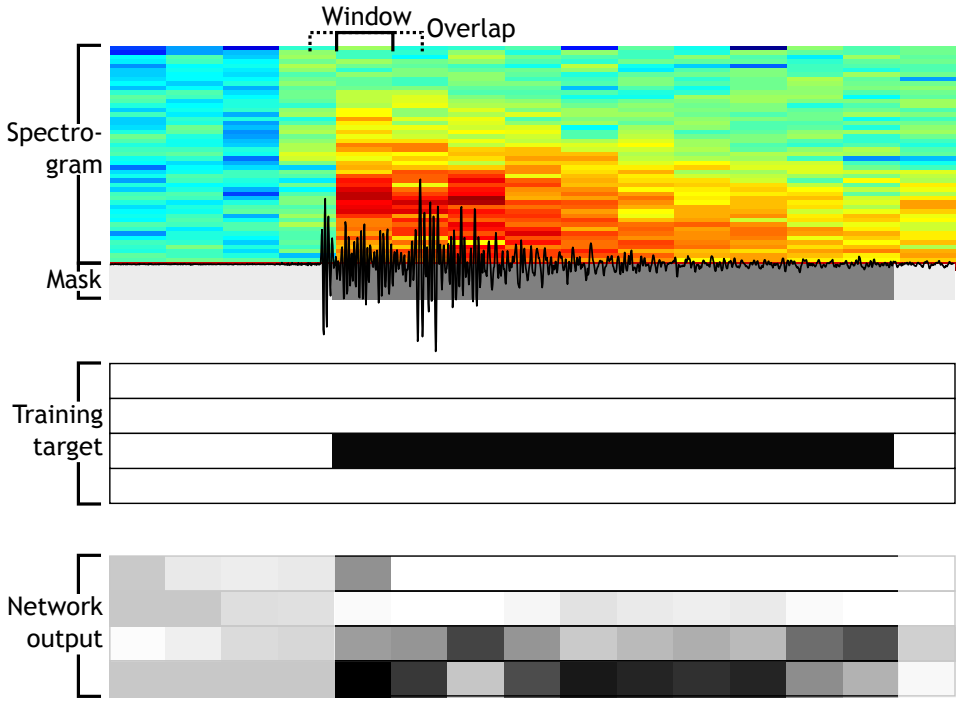

Figure 2.2. Creation of spectrograms, masks, and targets. A spectrogram and mask are generated from each input waveform. The mask is copied into a training target matrix, with one row per class, for use in training a neural network. Actual output from a neural network is shown for comparison—with a perfectly trained network, the network output would exactly match the training target. Network output outside the time extent of the mask is ignored. This figure shows only one spectrogram for a single event at one station; spectrograms are actually calculated from every input waveform, and concatenated as shown in Figure 2.3.

very deep or very distant earthquakes, where durations can be several minutes and critical frequencies less than 1 Hz, we would choose longer windows. This would improve resolution at low frequencies, but reduce temporal resolution.

In addition to a spectrogram, an earthquake mask is calculated for each waveform. The mask corresponds to the portion of the waveform (and the spectrogram) corresponding to an event. The mask is used to associate each time step of the spectrogram with a training target. Each training target vector has the same number of elements as event types; all are set to 0 except for the event type corresponding to the earthquake, which is set to a fixed non-zero value (Figure 2.2). The mask can be calculated by any appropriate means, such as a short-term-average/long-term-average threshold. In our test cases we found that the

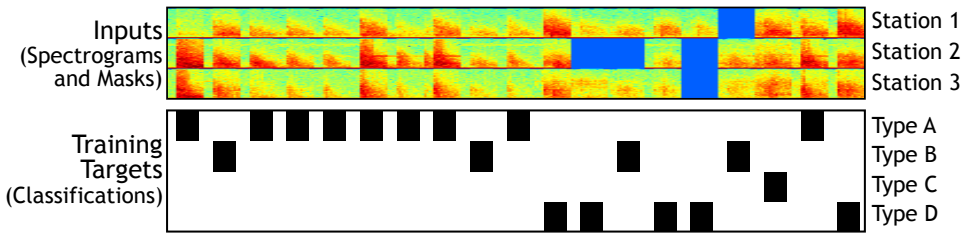

Figure 2.3. Combined spectrogram and target vector time series. Spectrograms are aligned across stations by phase arrival time. Empty spectrograms (i.e. blue boxes) indicate that no arrival was present in the catalog for those earthquakes at those stations. This figure contains spectrograms for 19 earthquakes; typically the spectrogram and target matrices will be much longer, if the input data contains hundreds or thousands of earthquakes.

exact length of the mask was unimportant, so we simply used a fixed length beginning at the initial arrival time of each event.

The mask is also used in the post-processing stage (Section 2.6) to calculate the final output event type for each spectrogram. Neural network output outside the time extent of the mask is ignored during this stage.

After spectrograms and target vectors are generated from each waveform, they are each stacked vertically by station and concatenated horizontally by event to form two large vector time series (Figure 2.3) to present to the neural network for training.

2.5 Neural Networks

An artificial neural network is at the core of the WC2 classification procedure. As described in Section 2.2, the time-delay neural network (TDNN) that we use was originally developed to solve a problem in speech recognition. *Waibel et al.* [1989] used such a network to classify phonemes in human speech. Each phoneme in their dataset is presented to the network as a spectrogram; they present the data and train the network in such a way that the network is robust to differences in phoneme onset relative to spectrogram start time. This is accomplished in part by presenting multiple time-shifted copies of each spectrogram to the neural network. MATLAB's Neural Network Toolbox (NNT) includes functionality for a more generalized form of *Waibel et al.*'s time-delay network design. Instead of time-shifted data, the input is presented as a complete time series. A classification is made at every time step in the series (Figure 2.4).

Though the general structure of every time-delay neural network is the same, it is nec-

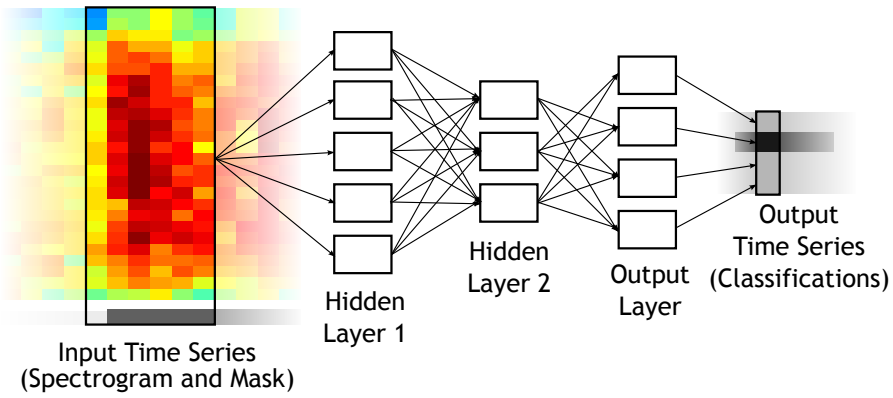

Figure 2.4. Schematic of a feed-forward time-delay neural network. In this example network, six seconds of spectrogram and mask values are passed through three layers of neurons to produce one second of classification output values. In this figure, part of only one spectrogram is shown, but in reality several spectrograms are horizontally and vertically concatenated, as in Figure 2.3.

essary to choose specific design parameters for each problem. In particular, the number of layers, the number of neurons within each layer, the activation function used by each neuron, and the number of delays (input time steps) must be chosen. It is also necessary to choose a training algorithm and to establish criteria for stopping training. All of these choices will affect both classification success and computational performance (i.e. speed of training and simulation). The best choices are dependent on the specific problem and datasets being used, and the goal of this study is not to present a specific set of best neural network parameters. The network parameters used in our three examples (described in Chapter 3) can be used as starting points, but for optimal classification success each user must perform his or her own experimentation to find the network that will best work for his or her classification problem. Some techniques for such experimentation are described in the Appendix.

Once the network parameters have been chosen, the network must be trained. Training is an iterative process whereby neuron weights are adjusted to minimize differences (typically mean squared error) between target network output and actual network output for a given set of input training data. Many sophisticated training algorithms have been developed; in this project we are concerned mainly with the results and performance of trained networks, not the specific mathematical details of training algorithms. We chose to use the Scaled Conjugate Gradient backpropagation algorithm [*Møller*, 1993] for all neural

networks tested in this project (Section A.4). We also created a modified mean squared error function to handle the masked input and target values (Section A.3).

In practice, we found that networks with identical parameters trained with identical training data could have very different performance on identical testing data. This is due to the random starting weights with which the network is initialized. Thus it is useful to train multiple networks and pick the best one[3]. Examples of this procedure are given in Chapter 3.

2.6 Post-processing

The trained TDNN provides an output classification at every time step, so there are multiple classifications per event. This is useful in some applications (e.g. for real-time continuous monitoring), but in most cases a single classification per event is desired. In order to determine this single classification, the output time series is averaged over the duration of each event. Then the event type with the greatest corresponding average output value is selected to be the final event type (Figure 2.5).

2.7 Classification Performance

Once each event in a testing or validation set has been classified, these output classifications (the predictions) can be compared to the actual classifications (the targets). This comparison provides a measure of classification performance, and is required in order to decide which algorithms work well and which ones do not.

The full description of performance on a particular dataset can be given by a confusion matrix, which lists the count of each target-prediction pair. The confusion matrix in Table 2.1 indicates, for example, that 12 events of type A were correctly classified as type A, but 5 events of type A were incorrectly classified as type B. Zero events of type A were classified as type C. The recall for type A is thus $r_{\mathrm{A}} = \frac{12}{12+5+0} = 0.71$. In other words, 71% of the type A events in the dataset were correctly classified by the classifier. The average

[3]This introduces a subtle complication to the testing process. Ideally, the same testing data should not be used to both choose the final network and to provide a final objective measure of that network's performance. Once the network is chosen, a second, separate testing dataset ought to be used to evaluate its performance. However, establishing this additional testing dataset will further reduce the amount of data available for training. We believe the effect of omitting this step is minor, and indeed our final Spurr results, where completely novel data was in fact presented, were quite good (Table 3.8).

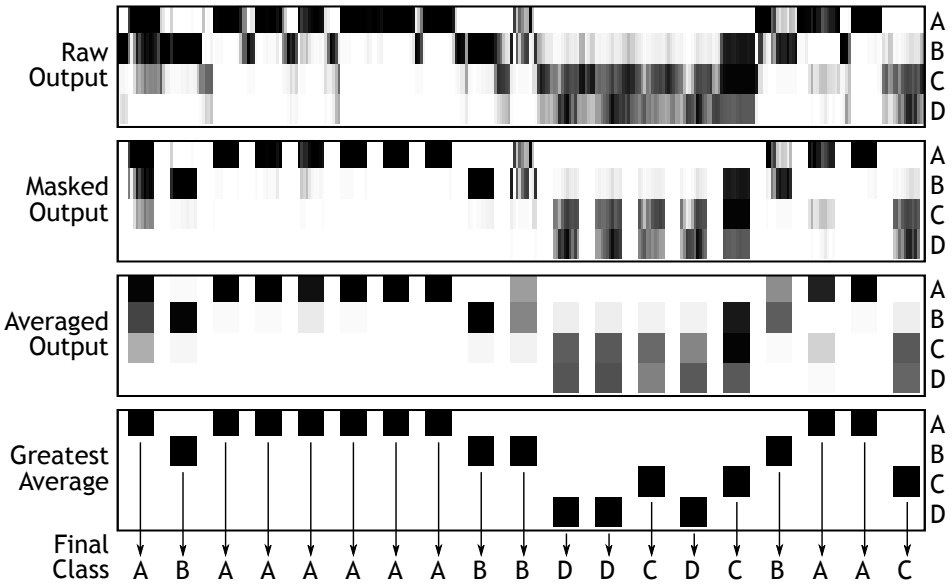

Figure 2.5. Average output values determine class. The raw output has earthquake masks reapplied, then the average output value within each class is calculated. The greatest average value indicates the final output class. This figure shows 19 earthquakes only; typically the raw output matrix will be much longer, if the input data contains hundreds or thousands of earthquakes (cf. Figure 2.3).

Table 2.1. Confusion matrix example. Rows correspond to the actual (target) class and columns correspond to the class predicted by the classifier. Elements on the main diagonal indicate correct classifications, and off-diagonal elements indicate incorrect classifications. For example, 12 events of type A are correctly classified as type A, but 5 events of type A are incorrectly classified as type B.

		Output			
		A	B	C	Totals
Target	A	12	5	0	17
	B	4	33	1	38
	C	0	0	16	16
Totals		16	38	17	71

recall $\bar{r}$ for the entire table is

$$\bar{r} = \frac{1}{3}(r_A + r_B + r_C) = \frac{1}{3}\left(0.71 + \frac{33}{4+33+1} + \frac{16}{0+0+16}\right) = 0.86 \tag{2.1}$$

Reducing a confusion matrix to a single elegant value is a source of much contention and many such measures have been proposed [e.g. *Liu et al.*, 2007; *Sokolova and Lapalme*, 2009]. For this project, we generally use average recall $\bar{r}$ as the figure of merit. Another measure, overall accuracy a, is the simplest and most intuitive—it is simply the fraction of correct classifications overall. In the above example, $a = \frac{12+33+16}{71} = 0.86$. The trouble with this measure is that classes with more members are weighted more heavily. For unbalanced datasets, overall accuracy may not be very informative. We discuss this in more detail in Sections A.1 and A.2.

Chapter 3
Data and Results

This chapter describes three case studies to which we applied our classification procedure: Uturuncu Volcano, Katmai Volcanic Cluster, and Mount Spurr. In each case we describe the data, the classifier and neural network parameters, and the results.

3.1 Uturuncu

Uturuncu, a volcano in southwest Bolivia, has been of recent scientific interest due to observed rapid uplift in the area, with deformation models indicating a deep magma source beneath the volcano [*Sparks et al.*, 2008]. Recent seismic studies have attempted to characterize local seismicity beneath the volcano and in the surrounding area [*Jay et al.*, 2012]. The seismic catalog from the ANDIVOLC/PLUTONS field campaign contains clean, high-quality data, and two distinct event types are readily identifiable. We chose to use these events to initially develop and test the classification procedure.

3.1.1 Data Description and Event Types

The ANDIVOLC catalog contains 1772 earthquakes recorded from May 2009 through April 2010 by 15 short-period seismometers surrounding Uturuncu (Figure 3.1). We visually examined every waveform and assigned one of two types to each earthquake: type V, volcano-tectonic events near the surface, and type D, deep slab earthquakes. Since we intended this to be a simple problem for basic testing of a procedure, we decided to omit events that did not fit into either of these types. Examples of these events are shown in Figure 3.2.

Of the 15 stations in the network, we chose four to use for this test: UTCA, UTCM, UTKH, and UTLA2. We chose the subset of earthquakes with P-wave arrivals picked at these stations from May 2009 through December 2009, and imported waveforms associated with these arrivals. This provided a total of 2893 waveforms associated with 921 earthquakes (note that most of the earthquakes did not have picked arrivals at all four stations).

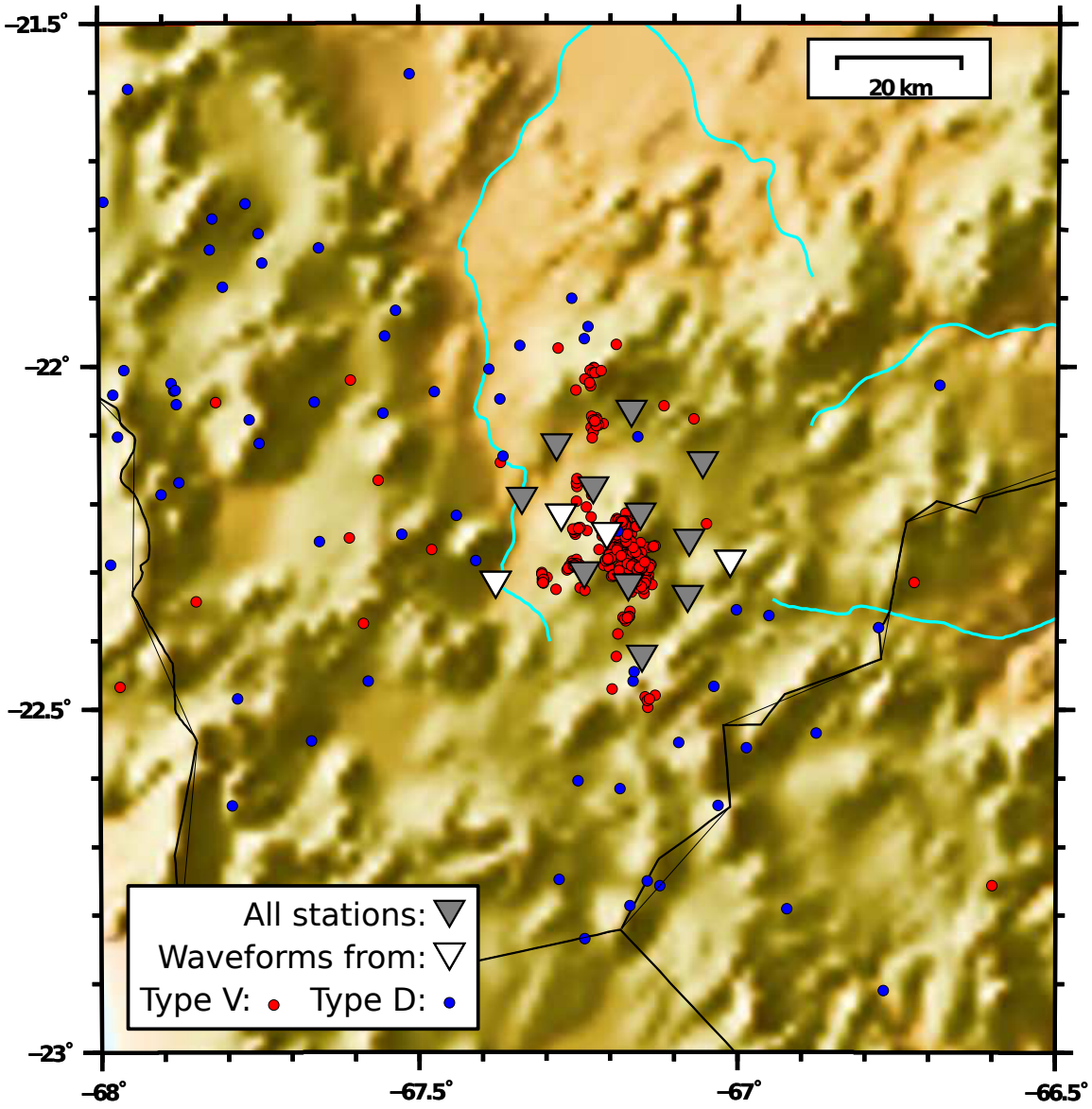

Figure 3.1. Map of Uturuncu network and earthquakes. Type V (volcano-tectonic) events in red are clustered near the volcano, and type D (deep) events in blue are scattered widely. Many type D events were located very distant to the volcano and are not shown on this map.

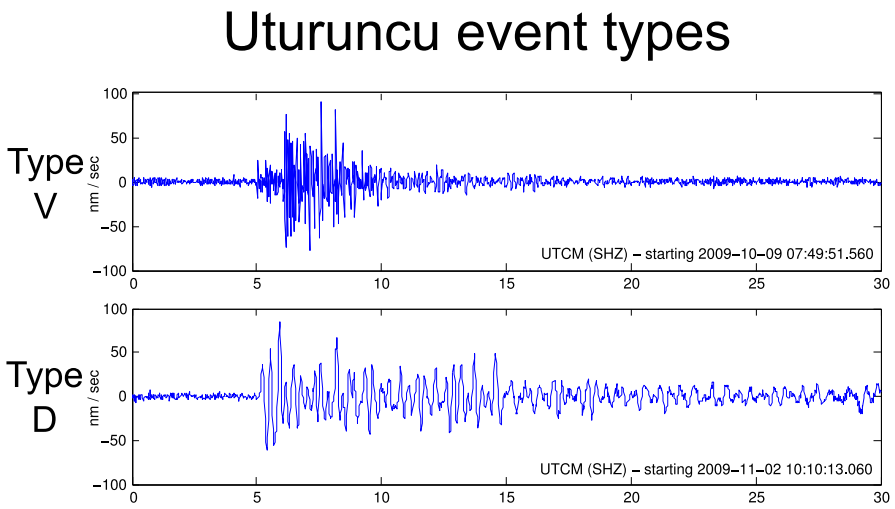

Figure 3.2. Examples of Uturuncu event waveforms. Type V events typically were high-frequency with distinct P and S phases. Type D events were low-frequency without distinct S phases.

3.1.2 Classifier Parameters

Each waveform was eight seconds long; two seconds prior to each arrival and six seconds following. The mask length was fixed at five seconds. Delays of zero through one second were used at the neural network's input layer, with no delays at subsequent layers. These time choices were deliberately limited. Most events in the dataset last longer than five seconds, so in most cases the mask (and indeed the entire waveform) did not fully cover the available data. Also, only two seconds' worth of data was used at a time by the neural network. Despite these limitations, the automatic classifier was very successful at distinguishing between type V and D events (Section 3.1.3).

Spectrograms were calculated with a window length of one second and a FFT (Fast Fourier Transform) length of at least one second. All four stations provide 50 Hz data, so 26 frequency bands were calculated from each (0 Hz, 0–1 Hz through to 24–25 Hz). Each second of input thus has 104 spectrogram values, plus four mask values (one per station), for a total of 108 input values per second. With delays of zero through one second, the neural network accepts a total of 216 input values at once.

For this Uturuncu dataset, a two-layer network was used, with three neurons in the

Table 3.1. Uturuncu test results. Confusion matrix for one trial with Uturuncu dataset. D=deep events, V=near-surface volcano-tectonic events. Overall accuracy is 0.996 (273 correct of 274), average recall is 0.994, average precision is 0.997.

		Output		
		D	V	Totals
Target	D	80	1	81
	V	0	193	193
Totals		80	194	274

first layer and two neurons (one per event type) in the second layer. A hyperbolic tangent activation function was used for all neurons (Section A.6). A scaled conjugate gradient function was used for training the network and the mask-adjusted mean squared error (Section A.3) was used to calculate performance during training. The network was configured to stop training after 2.5% mean squared error was reached, or after 1000 training iterations.

3.1.3 Performance and Remarks

We ran 30 training–testing trials with identical parameters and identical data. Most trials took less than two seconds to complete on a desktop PC. Testing data results for one trial are shown in Table 3.1. Summary statistics for all trials are given in Table 3.2. We see near-perfect results in every single trial. This reflects the major differences between the two event types; the neural network was successfully trained to distinguish them with ease. We note however that we consider this data set trivial to classify (recall Figure 3.2), which is part of why we chose to use it—it is the "easy" example, used to develop and fine-tune the classification procedure before applying it to more difficult problems. In practice it is probably not necessary to use a neural network for this problem.

3.2 Katmai

The Katmai Volcanic Cluster contains several active volcanoes monitored by the Alaska Volcano Observatory (AVO) using a distributed seismic network. We divided the area into four regions representing earthquakes at four different volcanoes: Trident, Martin, Katmai, and Snowy (Figure 3.3). The goal for this dataset was to automatically determine which

Table 3.2. Uturuncu trial performance. Summary statistics for all 30 trials. The best networks had perfect performance (accuracy and recall of 1) on the test dataset, and the worst network had overall accuracy of 0.960 and average recall of 0.932.

Result	Mean	Std. Dev.	Best	Worst
Training Time (s)	1.1	0.9	0.5	4.9
Training Iterations (Epochs)	21	17	8	93
Overall Accuracy (Training Data)	0.990	0.007	1.000	0.977
Average Recall (Training Data)	0.986	0.010	1.000	0.964
Overall Accuracy (Testing Data)	0.985	0.011	1.000	0.960
Average Recall (Testing Data)	0.978	0.017	1.000	0.932

volcano an earthquake originates from. This may seem redundant since AVO already locates these earthquakes, but there are two good reasons to do this.

First, if an earthquake cannot be located by traditional means (for example, if it is recorded on too few stations), automatic classification could still provide a general idea of the location based on the waveform(s) alone.

Second, classifying based on location allows the WC2 procedure to be tested on a completely objective dataset. With the Uturuncu dataset described previously, we had to decide on a classification scheme and decide which class each earthquake belonged to. Though this is the most commonly expected use of the procedure, it fundamentally lacks some scientific rigor when measuring classification performance. There is always a question of whether our subjective classification decisions affect the performance of the procedure. In contrast, an objective location-based classification scheme, such as that used in this Katmai dataset, allows us to evaluate the WC2 procedure on its merits alone, without possible influence from manual classification decisions.

3.2.1 Data Description and Event Types

We divided the Katmai area into four geographic regions, corresponding to seismicity roughly associated with four volcanoes: Trident (T), Martin (M), Katmai (K), and Snowy (S) (Figure 3.3). We selected one year's worth of earthquake origins located within these regions from the AVO catalog, spanning 08/01/2009 through 07/31/2010. We also selected arrival information and determined the five stations with the most picked arrivals associ-

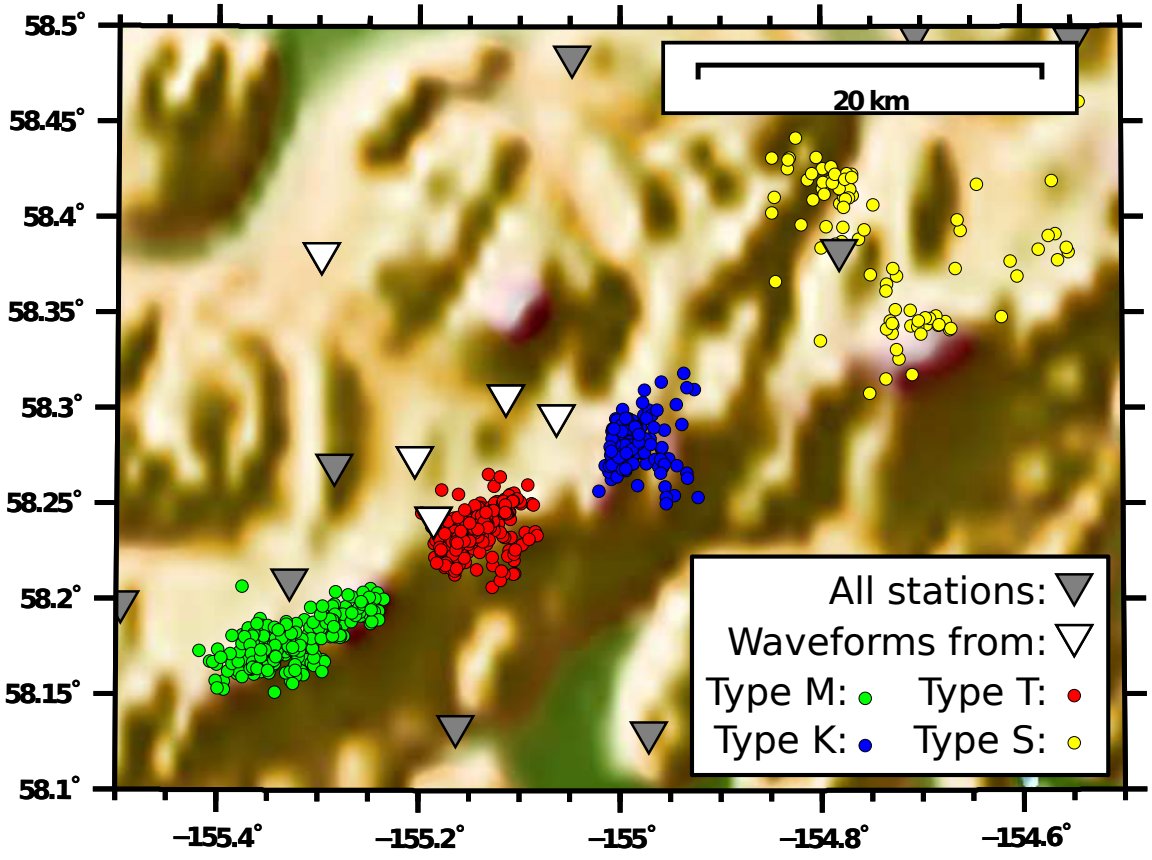

Figure 3.3. Katmai dataset map. Map shows all earthquakes within each defined region from 08/01/2009 through 07/31/2010. Earthquakes outside these regions are not shown.

ated with these earthquakes: KBM, KCE, KCG, KVT, and KAKN. Finally, we imported all waveforms associated with these arrivals, for a total of 4058 waveforms associated with 966 earthquakes (note that most of the earthquakes did not have picked arrivals at all five stations).

3.2.2 Classifier Parameters

Each waveform was 14 seconds long; four seconds prior to each arrival and 10 seconds following. The mask length was fixed at seven seconds. Delays of zero through four seconds were used at the neural network's input layer (with no delays at subsequent layers). These time choices ensure that enough of each event is represented in the data while maintaining an adequate buffer between each event.

Spectrograms were calculated with a window length of one second and a FFT length of at least one second. Four of the stations (KBM, KCE, KCG, KVT) provide 100 Hz data, thus 51 frequency bands resulted from the calculation (0 Hz, 0–1 Hz through to 49–50 Hz). The fifth station (KAKN) is digitized at 50 Hz, so 26 frequency bands were calculated (0 Hz, 0–1 Hz through to 24–25 Hz). Each second of input thus has 230 spectrogram values, plus five mask values (one per station), for a total of 235 input values per second. With delays of zero through four seconds, the neural network accepts a total of 1175 input values at once.

For this Katmai dataset, a two-layer network was used, with six neurons in the first layer and four neurons (one per event type) in the second layer. A hyperbolic tangent activation function was used for all neurons (Section A.6). A scaled conjugate gradient function was used for training the network and the mask-adjusted mean squared error (Section A.3) was used to calculate performance during training. The network was configured to stop training after 5% mean squared error was reached, or after 1000 training iterations.

3.2.3 Performance and Remarks

We ran 30 training–testing trials with identical parameters and identical data. Most trials took less than 10 seconds to complete on a desktop PC. Testing data results for one of the best trials are shown in Table 3.3. Summary statistics for all trials are given in Table 3.4. This test was also quite successful, with average recall of 0.89, and only 28 of 263 events

Table 3.3. Katmai test results. Confusion matrix for one trial with Katmai dataset. T=Trident, M=Martin, K=Katmai, S=Snowy. Overall accuracy is 0.89 (235 correct of 263), average recall is 0.87, average precision is 0.89.

		Output				
		T	M	K	S	Totals
Target	T	82	6	1	0	89
	M	5	96	0	1	102
	K	7	1	33	4	45
	S	1	1	1	24	27
Totals		95	104	35	29	263

Table 3.4. Katmai trial performance. The best trial had overall accuracy of 0.90 and average recall of 0.87 on the test dataset; the worst trial had overall accuracy of 0.81 and average recall of 0.70.

Result	Mean	Std. Dev.	Best	Worst
Training Time (s)	8.9	5.0	5.5	31
Training Iterations (Epochs)	69	40	41	251
Overall Accuracy (Training Data)	0.89	0.01	0.91	0.87
Average Recall (Training Data)	0.79	0.04	0.87	0.71
Overall Accuracy (Testing Data)	0.86	0.02	0.90	0.81
Average Recall (Testing Data)	0.76	0.04	0.87	0.70

classified incorrectly. Performance was worst on type K events (recall of 0.73), which the network had a slight tendency to classify as type T.

These results indicate that there is strong content within each waveform indicating its source region. This could be a combination of path effects and actual source mechanism differences at each volcano. We discuss path and source effects further in Section 4.3.

3.3 Spurr

Mount Spurr is a stratovolcano on the northeastern end of the Aleutian arc. From 2004 to 2006 it showed signs of volcanic unrest, including increased seismicity and deep long-

period events [*De Angelis and McNutt*, 2005; *Coombs et al.*, 2005]. Mount Spurr is actively monitored by the Alaska Volcano Observatory (AVO) with a network of permanent short-period and broadband seismometers. As part of the seismic monitoring process, earthquakes near the volcano are located and have an event type assigned by an AVO staff seismologist. Continuous waveform data has been stored from 2003 onward.

Whereas the Katmai dataset contains artificial event types based on earthquake location, the Spurr dataset uses the actual event types assigned by AVO analysts. The Spurr dataset also spans a longer period of time, and is subject to station outages and differing instrumentation. In other words, Mount Spurr allows us to test our procedure on the most realistic type of seismic data applicable to volcano monitoring, and our test simulates a real life production environment. Good classification performance on this dataset would make our procedure useful as a component of the monitoring process.

3.3.1 Data Description and Event Types

From the AVO catalog we selected nine years of earthquake origins within 25 km of the Mount Spurr summit, spanning 1/1/2003 through 12/31/2011. Each had been classified by an AVO analyst; we included only those which were classified as type A or B. These earthquake types correspond to the traditional volcano-tectonic (type A) and long-period (type B) events used in volcano seismology. We then split the type B earthquakes into two types: those located above 20 km deep (with respect to sea level), and those located below 20 km deep. We renamed the deep type B events to type C.

The split at 20 km establishes a third 'deep-LP' event type, which manifested more frequently during periods of unrest in 2004–2006 and in 2007. Figure 3.4 is a map of these earthquakes at Mount Spurr, colored by event type. Figure 3.5 shows the same earthquakes plotted by depth over time, with concentrations of type C events in 2004, 2005, and 2007 clearly visible. Figure 3.6 shows a selection of waveforms of each type as recorded at station SPBG.

We selected four stations to use for this classification test: CRP, BGL, SPBG, and SPCR. Waveform data from CRP and BGL is available for the entire time period, whereas SPBG and SPCR were installed in 2005. Of the 9629 earthquakes that were located in this time period, 4734 were type A, B, or C events that had recorded arrivals at at least one of the four chosen stations. A total of 8436 waveforms associated with these 4734 earthquakes were imported.

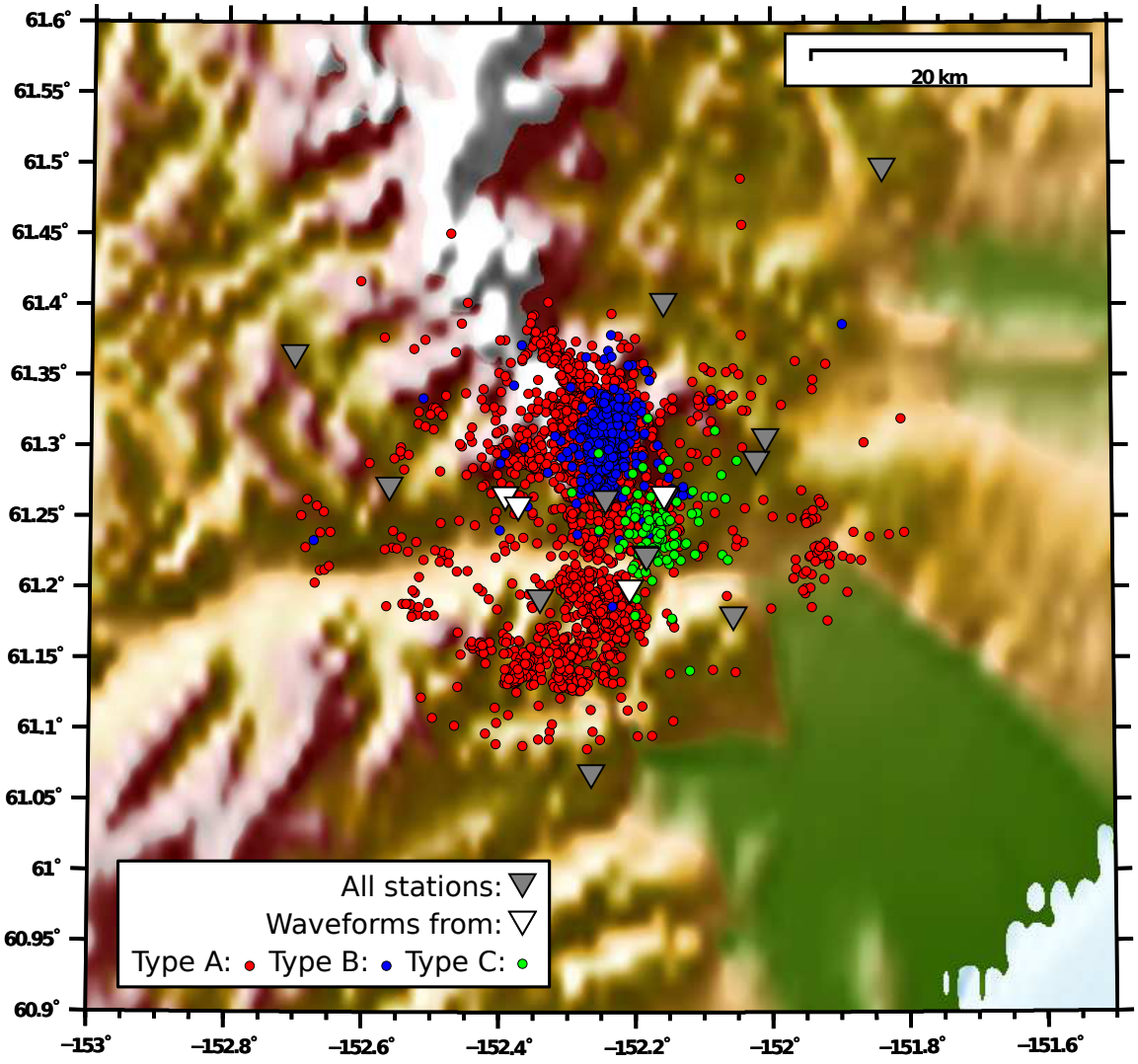

Figure 3.4. Spurr dataset and network map. Map shows all type A, B, and C earthquakes from 2003 through 2011 within 25 km of the Mount Spurr summit, with seismic stations marked.

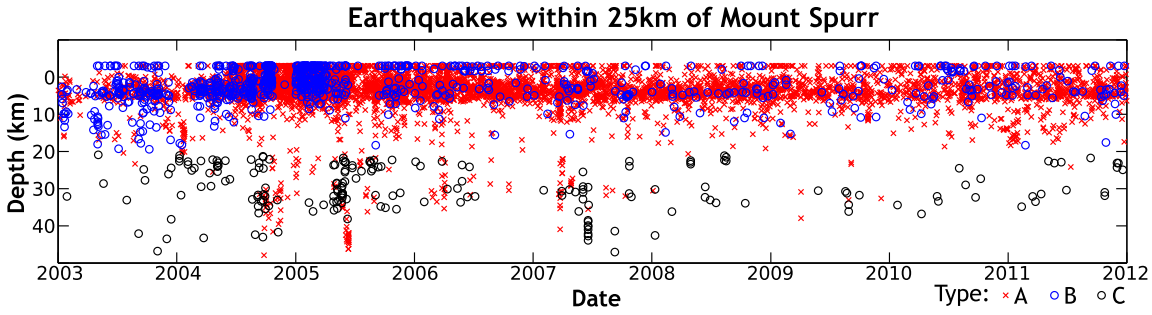

Figure 3.5. Depth-time plot of Spurr earthquakes. Depth vs. time plot of all type A (red x), B (blue circle), and C (black circle) events at Mount Spurr from 2003 through 2011. Clusters of type C events (deep LP events) are visible in 2004, 2005, and 2007.

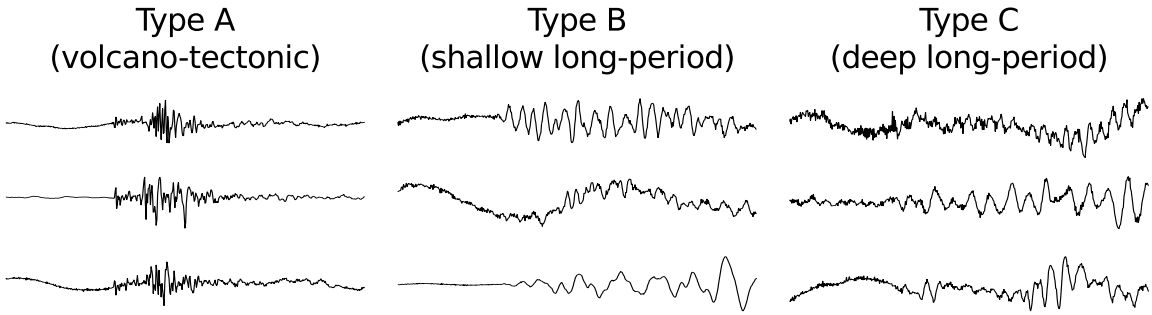

Figure 3.6. Spurr waveform examples. Examples of type A, B, and C events recorded at station SPBG. These events were randomly selected from the data; each waveform segment is 10 seconds long and amplitudes have been normalized. Type A events tend to have higher frequency content and distinct P and S arrivals. Type B and C are lower frequency and without distinct S arrivals. There are no immediately obvious differences between type B and C events.

3.3.2 Classifier Parameters

Each waveform used was 10 seconds long: three seconds prior to each arrival and seven seconds following. The mask length was fixed at four seconds. Delays of zero through three seconds were used at the neural network's input layer (with no delays at subsequent layers). These time choices ensure that enough of each event is represented in the data while maintaining an adequate buffer between each event.

Spectrograms were calculated with a window length of one second and a FFT length of at least one second. Two of the instruments (CRP, BGL) were sampled at 100 Hz, thus 51 frequency bands resulted from the calculation (0 Hz, 0–1 Hz through to 49–50 Hz). The two other instruments (SPBG, SPCR) were sampled at 50 Hz, so 26 frequency bands were calculated (0 Hz, 0–1 Hz through to 24–25 Hz). Each second of input thus has 154 spectrogram values, plus four mask values (one per station), for a total of 158 input values per second. With delays of zero through three seconds, the neural network accepts a total of 632 input values at once.

For this Spurr dataset, a two-layer network was used, with sixteen neurons in the first layer and three neurons (one per event type) in the second layer. A hyperbolic tangent activation function was used for all neurons (Section A.6). A scaled conjugate gradient function was used for training the network and the mask-adjusted mean squared error was used to calculate performance during training. The network was configured to stop training after 2% mean squared error was reached, or after 1000 training iterations. We split the input dataset in two by randomly sampling it: 65% of the waveforms went into training the neural network, and the other 35% were used to test the trained network. The Spurr trials were run on a different PC from the Uturuncu and Katmai trials, but it had similar specifications and capabilities such that training times were not significantly affected.

3.3.3 Performance and Remarks

As with the Katmai dataset, we initially ran training-testing trials on the entire Spurr dataset. The dataset and the neural network size were both slightly larger (and perhaps more challenging data for the network), so the training process took longer. However, because of the disparate numbers of each event type, the initial results were unacceptable. Most type A and the majority of type B events in the testing dataset were successfully

Table 3.5. Spurr initial test results. Confusion matrix for one trial with Spurr dataset. A=volcano-tectonic, B=long-period (< 20 km), C=long-period (> 20 km). Overall accuracy is 0.947 (1555 correct of 1642), average recall is 0.682, average precision is 0.797.

		Output			
		A	B	C	Totals
Target	A	1450	15	8	1473
	B	45	93	0	138
	C	15	4	12	31
Totals		1510	112	20	1642

classified, but the recall of type C events was poor. An example trial result is shown in Table 3.5.

The explanation for this behavior is that the neural network training process attempts to minimize overall classification error, regardless of the number of inputs of each class. Here, even though just 38.7% of the type C events were correctly identified, the overall accuracy is still quite good, at 94.7%.

Dealing with unbalanced data is discussed in Section A.2. We attempted another set of 15 trials on this dataset, but this time we balanced the data by limiting the type A and B events to 200 each, chosen randomly from the entire set. If the number of type A events is comparable to the numbers of types B and C events in the training data, the training process will treat each event type more equally when it minimizes classification error. And indeed, the results were much better, as shown for an example trial in Table 3.6. Summary statistics for all 15 trials are given in Table 3.7.

As one last test, we applied one of the trained networks described above (with results in Table 3.6) to a complete, unbalanced Spurr dataset (less the training data itself). Performance was comparable to that of the balanced dataset, with average recall of 0.81. Complete results are shown in Table 3.8. Compared to the original unbalanced trial, the recall for type C is still low ($\frac{43}{63} = 0.68$), but the confusion was mainly between type B and type C events (not type A events as in Table 3.5). Since types B and C represent an arbitrary depth-based split of the original AVO type B classifications, this confusion is expected, and suggests that there isn't a clear distinction between deep and shallow LP events in all cases, especially close to our chosen 20 km boundary.

Table 3.6. Spurr balanced test results. Confusion matrix for one trial with balanced Spurr dataset. A=volcano-tectonic, B=long-period (< 20 km deep), C=long-period (> 20 km deep). Overall accuracy is 0.864 (146 correct of 169), average recall is 0.857, average precision is 0.839.

		Output			
		A	B	C	Totals
Target	A	49	2	3	54
	B	6	72	6	84
	C	2	4	25	31
Totals		57	78	34	169

Table 3.7. Spurr balanced trial performance. The best network had overall accuracy of 0.876 and average recall of 0.858 on the testing data; the worst network had overall accuracy of 0.817 and average recall of 0.810.

Result	Mean	Std. Dev.	Best	Worst
Training Time (s)	35.5	8.1	28.7	62.8
Training Iterations (Epochs)	528	132.7	425	984
Overall Accuracy (Training Data)	0.981	0.004	0.985	0.973
Average Recall (Training Data)	0.976	0.007	0.984	0.963
Overall Accuracy (Testing Data)	0.857	0.013	0.876	0.817
Average Recall (Testing Data)	0.846	0.013	0.858	0.810

Table 3.8. Spurr balanced test results on unbalanced data. Confusion matrix for one trial with network trained with balanced data, applied to a complete unbalanced Spurr dataset. A=volcano-tectonic, B=long-period (< 20 km deep), C=long-period (> 20 km deep). Overall accuracy is 0.895 (2768 correct of 3093), average recall is 0.807, average precision is 0.624.

		Output			
		A	B	C	Totals
Target	A	2501	207	53	2761
	B	29	224	16	269
	C	3	17	43	63
Totals		2533	448	112	3093

Chapter 4
Discussion

4.1 Application to Real-Time Monitoring

In Section 1.1 we discussed the motivation to use automatic classification systems on large pre-existing data sets. This project achieved that goal, particularly with the Spurr data set, where over 4000 earthquakes were successfully classified using more than 8000 individual waveforms as input to a trained neural network. However, we also discussed a second motivation, of automatic classification as a component of a real-time volcanic monitoring system. Though we used real data to test our procedure, we did not attempt to integrate it into the monitoring workflow, nor even simulate how it would perform when presented with novel waveforms sequentially in time. This would be a logical next step for this research, and in particular would require a way to ensure that the neural network remains adequately trained as time progresses and as the seismic character of a volcano evolves.

4.2 Neuron Weight Analysis

An additional area to investigate would be the numerical description of a trained neural network, and how the neuron weights relate to the input data. In Section 2.1, we described a trained classifier as a "black box". This has remained effectively so, even though we know that we can mathematically describe exactly what is happening inside this box. We have proven that our procedure works well on the three data sets that we used, but this was ultimately an empirical test. We have not addressed the core of the algorithm: what exactly is the neural network "seeing" within the input data that allows it to classify waveforms correctly?

The trouble is that it is very difficult to interpret a large set of neuron weights. For example, in Section 3.2.2 we described the Katmai neural network as having 1175 input values, a six-neuron hidden layer, and a four-neuron output layer. This trained neural network is thus described by 7084 individual weight and bias values. Because of the unguided nature of training and the random initial values, there is no way to predict what these actually might mean. They will be different every time the network is trained. Many of these values may draw strongly from meaningless parts of the input data, but cancel each other out when the final output is calculated.

Waibel et al. [1989] did address this issue in their study of human phonemes, and were able to pick out specific frequency-domain features corresponding to different phonemes

using a specialized Hinton diagram [*Hinton et al.*, 1986] of their trained network's neuronal weights. Figure 4.1 is a highly annotated and rearranged form of Hinton diagram for one of the Uturuncu networks trained in Section 3.1. With some careful analysis (Section 4.2.1), we can see that spectrograms with strong content below 7 Hz will generally be classified as type D, and spectrograms with strong content above 7 Hz will be classified as type V.

4.2.1 Interpretation of Uturuncu Hinton Diagram

Neuron 2.1 has a strong positive influence from 1.1 and a strong negative influence from 1.3. Neuron 1.1 has a strong positive influence from bands 0–7 Hz and a strong negative influence from bands 7–22 Hz, neuron 1.3 has a strong negative influence from bands 0–7 Hz and a strong positive influence from bands 7–22 Hz. So in the end, neuron 2.1 has a strong positive influence from bands 0–7 Hz in both neurons 1.1 and 1.3. Thus if those bands dominate the input spectrogram, the output will be type D.

Conversely, neuron 2.2 has a strong negative influence from neuron 1.1 and a strong positive influence from neuron 1.2. Neuron 1.2 is weakly negative in both frequency bands; it will not really affect the output. So in the end, neuron 2.2 has a strong positive influence from bands 7–22 Hz, and if those bands dominate the input spectrogram, the output will be type V.

This matches with what we expect: type V represents high-frequency volcano-tectonic earthquakes, and type D represents low-frequency deep slab earthquakes. The weights within the trained neural network tell us that the cutoff is at 7 Hz. To confirm this independently, we calculated the mean instantaneous frequency [*Taner et al.*, 1979] of each waveform, and plotted the distribution for each type separately (Figure 4.2). We can see that 7 Hz indeed reasonably differentiates the frequency distributions of type V and D events.

Though we were able to "look inside" the neural network above, it should be obvious that this will not be feasible for networks much larger than this, and that it will be challenging to identify anything but the most basic features within the weights. Ultimately we believe that interpreting the numerical structure of a neural network is not a necessary task, though in simple cases it can provide genuine insight into features of the input data.

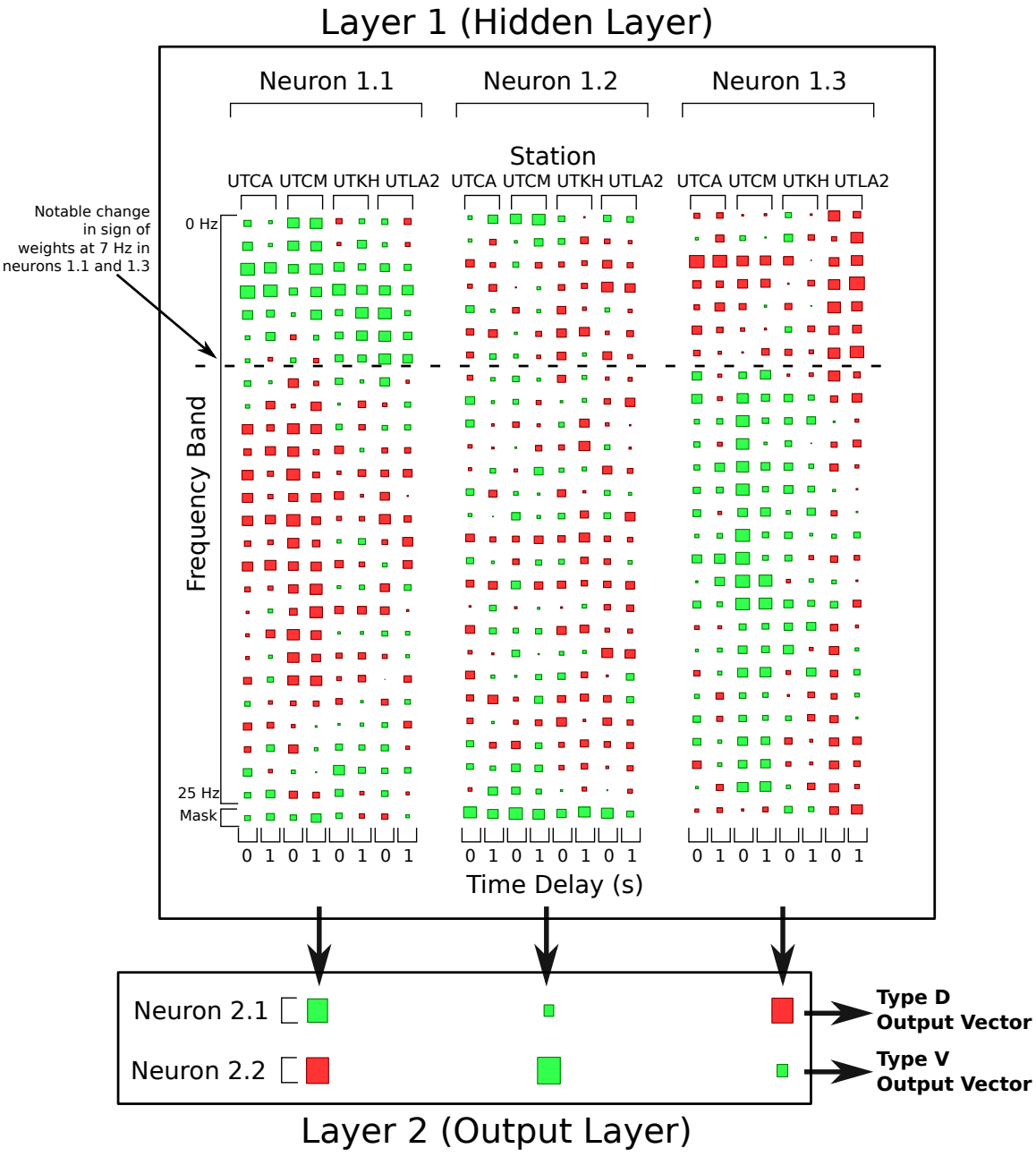

Figure 4.1. Hinton diagram for Uturuncu neural network. Each box represents an individual weight within the trained network; green is positive, red is negative, and the size of the box indicates magnitude. These are only the static weights themselves; the actual output is determined by multiplying input values by these weights, as described in Section 2.2 and Figure 2.1. The time delays represent the position within the 2-second sliding window associated with each weight. By following the weights through the network (Section 4.2.1), we can see that spectrograms with strong content below 7 Hz will generally be classified as type D, and spectrograms with strong content above 7 Hz will be classified as type V.

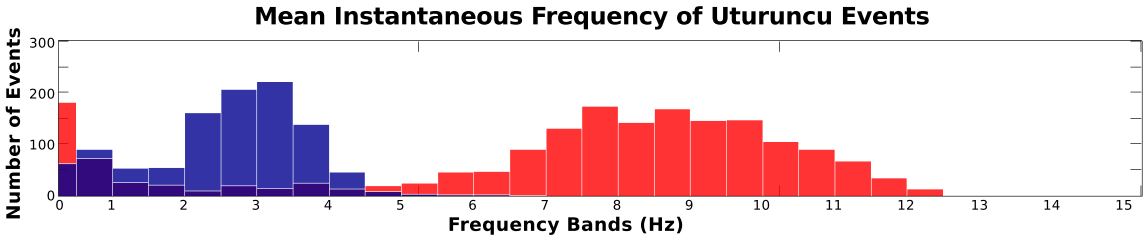

Figure 4.2. Frequency distribution of type V and D events. Type V events are shown in red, type D events are shown in blue.

4.3 Source and Path Effect Considerations

Our classification procedure does not explicitly account for any path effects, Earth structure, or site characteristics. However, these effects can be eliminated implicitly by the empirical nature of our approach. The neural network learns to classify earthquakes based on how they present at each individual station, *after* being subjected to any path and site effects. Features common to different earthquake types will be diminished during training, because they do not provide any basis for classification.

This argument is further strengthened by the fact that we present spectrograms from different stations at different positions within the input data. Each station has its own set of neural network weights associated with it, which will learn to individually account for any peculiarities at that station.

That said, our method is not limited to detecting differences in source characteristics. The classification of our Katmai data (Section 3.2) was ideally based entirely on path effects. We wanted to determine the approximate location of each earthquake regardless of the source mechanism and earthquake type. We accomplished this by training with a variety of earthquake types at each location. The neural network learned to emphasize the differences in how earthquakes from different locations presented at each station.

Ultimately we return to what we discussed in Section 2.1: some basis is required within the waveforms to differentiate between the different earthquake types. This is true no matter what the types actually represent.

A key limitation of this approach is that classification ability is fundamentally linked to site and path characteristics. Suppose we tried to use a previously trained network to classify earthquakes recorded at a brand new station: it would most likely fail miserably. We have mitigated this problem by classifying based on data from multiple stations; if a new station is added there can be some time overlap before we eventually retrain the net-

work using the new station. But suppose Mount Spurr had a particularly violent eruption that catastrophically destroyed its entire seismic monitoring network—we would not be able to classify earthquakes at a newly installed station, despite having over a decade of data from the destroyed seismic network. For some classification problems, we could reduce this risk by removing site and path effects before presenting waveforms to the neural network.

Sometimes path effects will be so significant as to eliminate differences in source features. Suppose we attempt to use a very distant station to classify volcanic earthquakes; due to attenuation effects, high frequency content will be lost, and all waveforms recorded at that station may look similar. In this case, the neural network would not be able to use data from that station to distinguish between earthquake types. If other nearby stations are also included in the input data, the network may still be able to perform a successful classification, but we would see that the neuron weights associated with the distant station were diminished. This is a potentially dangerous situation—we assume that our classification procedure has redundancy because it uses multiple stations, but we have no explicit way to identify which stations are actually being used, unless we are able to interpret a Hinton diagram or perform sensitivity testing.

4.4 Alignment Errors in Training Data

In our classification procedure we begin with waveform segments associated with specific events in a previously built earthquake catalog. This is in part so that we can align waveforms from different stations based on P wave arrival time, and so that we can assign a discrete class to each event.

What would happen if the P picks for a particular event are incorrect? If this occurs within the training data, the network will try to learn to classify noise (i.e. waveforms not associated with earthquakes) as actual event types. If this happens for a small number of events, the effect will be negligible, because many more events exhibit the correct behavior and thus have a stronger influence during the training process. If a significant portion of events have bad P picks, the effect will depend on how offset the picks are compared to the mask length.

For example, suppose the mask is 10 seconds long, and P picks are consistently early by 5 seconds, such that the first 5 seconds of each masked segment represents non-earthquake noise. This effectively means that 50% of the data going into training has no basis to dif-

ferentiate between earthquake types. However, the other 50% is fine. This is likely still enough data to properly train the network. In some ways this is similar to the issue of balanced and unbalanced data discussed in Section 3.3.3—the 50% of "good" data still has a strong enough effect on the mean squared error to successfully train the network. On the other hand, if the P picks are early by 9 seconds, only 10% of the waveform data presented represents actual earthquakes. This is too small of a proportion for training to be successful.

4.5 Numerical Class Output

Recall that the raw output from a neural network in this procedure is not a single discrete class name, but rather a numerical value for each class per second (Figure 2.5); we determine the final class by averaging the output and selecting the greatest value. However, there are other possible ways to interpret this output.

Since the network is trained to produce output of either 1 or 0, and the output neurons use a sigmoid activation function (Section A.6), it is unlikely that values falling somewhere in between (e.g. 0.5) represent anything numerically meaningful. They may be produced by an event with only some target features, but we do not have a way to quantify this. A more useful approach would be to look at how the values change over the course of an event. For example, if an event produces mainly type A volcano-tectonic output near the start and switches to type B long-period output part way through, this could indicate a hybrid type event. Simple averaging of the output values would not pick up on this.

An event may also be presented that does not fit into one of the target types. In this case, the output values should all be close to 0, and taking the greatest average would not provide meaningful information. It would be possible to use an output threshold to identify events as "unclassified" or with "unknown type".

4.6 Role of Continuous Waveform Input

In our classification procedure we begin with waveform segments associated with specific events in a previously built earthquake catalog. However, the actual input to the neural network is simply a long matrix of spectrograms, with event boundaries only indicated by the mask. The network processes this input along a sliding window, and does not fundamentally need complete, discrete events in order to come up with classification outputs.

In theory, it would only take slight modification of the procedure to incorporate actual

event detection. During training, instead of masking out the data between earthquakes, it could be assigned an explicit type such as “non-event”. Then the network would be trained to produce event boundaries and classifications at the same time.

This would still require some kind of event detection procedure to prepare the initial training data, such as a a short-term versus long-term average amplitude. P arrivals would also no longer be aligned, which might affect the ability to correctly determine event types. However, these are tractable problems. Artificial neural networks are very powerful tools, and we have only developed one specific approach to classifying earthquakes. There are many possible options for future work.

Chapter 5
Conclusions

In this thesis we developed a procedure to classify seismic waveforms using artificial neural networks, and tested this procedure on three different volcanic earthquake data sets. On all three data sets, the procedure was able to correctly classify the vast majority of earthquakes presented to it, as summarized in Table 5.1. We have thus successfully introduced a new tool for seismologists to use when classification is required. More generally, we have reinforced that artificial neural networks are a useful tool for seismic waveform analysis.

The inspiration and basis for the particular type of neural network used as well as the input data format (i.e. spectrograms as opposed to raw waveforms) was the work of *Waibel et al.* [1989], who used neural networks to classify phonemes in human speech. We successfully applied a simplified version of their method to classify seismic waveforms from three different volcanic regions. We obtained successful results using simple networks with a single hidden layer of less than 20 neurons (Chapter 3). Furthermore, we determined that there would be no improvement to the results by using larger or more complex networks (Section A.5). We determined the most appropriate training algorithm to use for this type of problem (Section A.4). And we discussed performance measurement, establishing average recall $\bar{r}$ as the most appropriate figure of merit for classification performance (Sections 2.7 and A.1).

We have proven that our procedure was effective on three varied input data sets. We have also provided a framework for future evaluation of neural network performance. If careful testing is performed with separate training and testing data sets, the effectiveness of a trained network can be trusted.

Table 5.1. Summary of performance from Uturuncu, Katmai, and Spurr. These are the mean average recall and mean overall accuracy from the testing trials described in Chapter 3.

Metric (mean of all trials)	Uturuncu	Katmai	Spurr (balanced)
Overall Accuracy	0.99	0.86	0.86
Average Recall	0.98	0.76	0.85

Appendix

A.1 Classification Performance

The confusion matrix introduced in Section 2.7 concisely describes the performance of a single classification trial, but its N^2 variables (N is number of classes) make it unsuitable for quantitative comparison of performance across many trials. The scalar measures F-score (F) and accuracy (α) summarize the information within the confusion matrix using single values and are suitable for quantitative performance comparisons. These measures are described here, based on the analysis of *Sokolova and Lapalme* [2009].

A.1.1 True and False Positives and Negatives

A two-class classification scheme is equivalent to a binary (positive-negative) classification scheme. Classification using a binary scheme may result in true positive (*tp*), false positive (*fp*), false negative (*fn*), and true negative (*tn*) examples, as shown in Table A.1. Confusion matrices will be represented hereafter in matrix form, with true/target classes represented by rows and output classes represented by columns. Thus the confusion matrix in Table A.1 is equivalently represented as

$$\mathbf{C} = \begin{pmatrix} tp & fn \\ fp & tn \end{pmatrix} \tag{A.1}$$

Multi-class confusion matrices can be broken out into multiple binary confusion matrices, indicating true and false positives and negatives with respect to each class. Summation is explicitly indicated. The variables c_{mn} refer to the number of examples of class m

Table A.1. Confusion matrix for binary classification.

		Output	
		Positive	Negative
Target	Positive	True positive tp	False negative fn
	Negative	False positive fp	True negative tn

classified as class n.

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N1} & c_{N2} & \cdots & c_{NN} \end{pmatrix} \tag{A.2}$$

$$\mathbf{C}_i = \begin{pmatrix} tp_i & fn_i \\ fp_i & tn_i \end{pmatrix} \tag{A.3}$$

$$= \begin{pmatrix} c_{ii} & \sum_{n=1}^{N} c_{in} - c_{ii} \\ \sum_{n=1}^{N} c_{ni} - c_{ii} & \sum_{n=1}^{N} c_{nn} - c_{ii} \end{pmatrix} \tag{A.4}$$

A.1.2 Accuracy

The most intuitive single performance measure is overall accuracy α_μ. This is simply the fraction of correctly-classified examples in the output.

$$\alpha_\mu = \frac{\sum_{n=1}^{N} c_{nn}}{\sum_{n=1}^{N} \sum_{m=1}^{N} c_{mn}} \tag{A.5}$$

Another measure, average accuracy α_M, weights each class equally by calculating accuracy for each binary matrix C_i and taking the average.

$$\alpha_\mathrm{M} = \frac{1}{N} \sum_{n=1}^{N} \frac{tp_n + tn_n}{tp_n + fn_n + fp_n + tn_n} \tag{A.6}$$

Both of these forms of accuracy only measure correctly classified examples (true positives and true negatives). This may not be adequate for measuring performance if the false positives and false negatives are important. In a volcano monitoring context, a false positive could be economically wasteful and a false negative could put lives at risk.

A.1.3 Precision, Recall, and F-score

Two additional measures, precision p and recall r take false classifications into account. In some fields precision and recall are referred to as producer's accuracy and user's accuracy

[e.g. *Liu et al.*, 2007]. They also have overall and average forms.

$$p_\mu = \frac{\sum_{n=1}^{N} tp_n}{\sum_{n=1}^{N} \left(tp_n + fp_n\right)} \tag{A.7}$$

$$r_\mu = \frac{\sum_{n=1}^{N} tp_n}{\sum_{n=1}^{N} \left(tp_n + fn_n\right)} \tag{A.8}$$

$$p_\mathrm{M} = \frac{1}{N} \sum_{n=1}^{N} \frac{tp_n}{tp_n + fp_n} \tag{A.9}$$

$$r_\mathrm{M} = \frac{1}{N} \sum_{n=1}^{N} \frac{tp_n}{tp_n + fn_n} \tag{A.10}$$

The final measure described is F-score F, the harmonic mean of precision and recall. This provides a balanced overall measure of classification performance, but does not include true negatives in binary classification (these are implicity included in the multi-class calculation, because true negatives within one class comprise true positives in other classes). F-score also has overall and average forms.

$$F_\mu = \frac{p_\mu r_\mu}{p_\mu + r_\mu} \tag{A.11}$$

$$F_\mathrm{M} = \frac{p_\mathrm{M} r_\mathrm{M}}{p_\mathrm{M} + r_\mathrm{M}} \tag{A.12}$$

In comparing the performance of classifiers, we chose to use average recall (r_M) as the figure of merit. Recall indicates the fraction of examples in a target class that are correctly classified in the output. In contrast, precision indicates the fraction of examples in an output class that are correctly classified. Section 2.7 includes a concrete calculation of average recall that may be more enlightening than Equation A.10. However, Equation A.10 does emphasize that false negatives, as opposed to false positives, are used in the calculation. Average F-score, while being the most comprehensive measure, dilutes the individual effects of false negatives and false positives. We believe that in a natural hazards context, false negatives ought to be weighted more heavily than false positives, and average recall provides this functionality. For applications outside of volcano monitoring or natural hazards, another measure listed above could be more appropriate.

A.2 Balancing Training Data

Standard neural network training methods work to reduce the overall error between classification outputs and targets associated with the training dataset. If the numbers of events of each class are approximately equal, each class will be treated equally during the training

process. However, if some classes have many more events, those classes will dominate the minimization. For example, if a training dataset contains 99 type A events and a single type B event, the neural network can classify all 100 events as type A and still obtain 99% overall accuracy and a low mean squared error. The network would have numerically good performance but be useless as a classifier. We see a real example of this in table 3.5 in section 3.3.3, which shows very poor classification performance for type B and C events, despite overall accuracy of nearly 95%. This particular classifier had an unwanted tendency to classify events as type A.

The simplest solution to this problem is to adjust training data to be more balanced, either by deleting events with disproportionately high quantities or by duplicating events with low quantities. We tried this for Spurr data in section 3.3.3, with good success. By eliminating most of the type A events in the training data, we trained a much more effective classifier (table 3.8). During the training process, each of the three classes had an approximately equal effect on the mean squared error, so the network did not skew toward one class in particular.

A more complicated solution is to adjust the training error function to compensate for unbalanced data. We remind the reader that the classification performance metrics discussed in section A.1 are intended to describe performance to humans and to compare performance between different classifiers; metrics such as F-score and average recall are not directly involved in training a neural network. A typical training error function is mean squared error, which treats each input data point equally.

It would be possible to adjust the mean squared error function to weight classes differently or to provide some other internal balancing procedure. We made one such adjustment to the standard function, to mask unimportant data (section A.3). In theory we could implement an additional change, for example to calculate mean squared error separately for each class and then calculate the average. But because of the complexity of this task (in part because the training algorithm also requires the *derivative* of the error function with respect to each data point) we did not attempt to implement this. Simple balancing of the input data as described above proved effective enough in our test cases.

A.3 Training Error Function

A neural network training algorithm requires an error function, used to calculate network performance and to adjust neuron weights during each iteration. A commonly used func-

tion is the mean squared error E, which is a measure of the difference between network output values (Y, in our case the actual output event types) and the training targets (T, the desired event types) over N data points.

$$E = \frac{1}{N}\sum_{n=1}^{N}(T_n - Y_n)^2 \tag{A.13}$$

E measures the overall error, and training can stop when it drops below a specified threshold. The more important value is the derivative E'_n with respect to each output value Y_n.

$$E'_n = -\frac{2}{N}Y_n \tag{A.14}$$

These derivatives are propagated backward through the layers of the network according to the specific training algorithm in use. Recall, however that we present non-earthquake data during training as well (Figure 2.3). This acts as a buffer between seismic events when multiple events are combined into a single input data matrix (thus preventing values from one earthquake spectrogram from affecting classification of another), but also skews the network toward zero output values. The training algorithm finds that E can be minimized by producing zeros from every input, which is obviously not what we want.

We solve this problem by using the earthquake mask to exclude the buffer values from the calculations of E and E'_n. If M represents the mask values, with $M_n = 1$ for earthquake data points and $M_n = 0$ for buffer data, the modified mean squared error $\hat{E}$ and its derivatives $\hat{E}'_n$ are as follows.

$$\hat{E} = \frac{1}{\sum M}\sum_{n=1}^{N}M_n(T_n - Y_n)^2 \tag{A.15}$$

$$\hat{E}'_n = -\frac{2}{\sum M}M_n Y_n \tag{A.16}$$

The effect of this change is that buffer data is essentially ignored entirely during the training process. This results in nonsense output anywhere the mask vectors are zero, but this output is ignored during the post-processing steps (Section 2.6) and does not affect the final output classes.

A.4 Training Algorithm

There are many algorithms that have been developed to train artificial neural networks. MATLAB's Neural Network Toolbox (NNT) provides access to several of these. There is no previously established best-choice algorithm for this type of problem and network

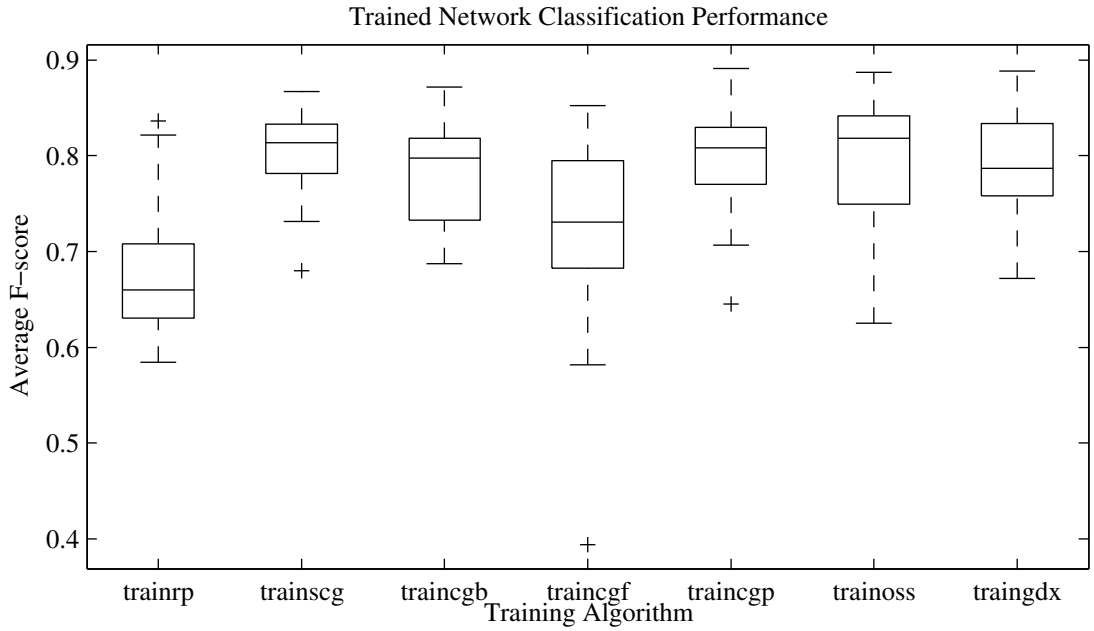

Figure A.1. Training algorithm performance. Box plot of average F-score for 30 trials run using each of 7 algorithms.

structure, so it was necessary to compare the performance of several algorithms to decide which one is best.

The primary goal here was to determine which algorithm produced networks with the best classification performance, while remaining capable of running on a standard lab PC. By training networks with the same structure and the same training data multiple times, performance distributions (Figure A.1) were calculated.

We established a baseline network structure consisting of one 6-neuron hidden layer and one 4-neuron output layer, with input delays of 0 through 6 seconds and no delays in the output layer. Both layers used hyperbolic tangent sigmoid activation functions. Katmai data was used for training. We trained 30 networks with each of 7 algorithms (210 trials total). Training was set to stop as soon as mean squared error dropped below 2.5% or after 2000 iterations.

Two efficient training algorithms available within MATLAB were omitted from this test. Levenberg–Marquardt backpropagation (`trainlm`) and BFGS quasi-Newton backpropagation (`trainbfg`) exceeded our computer memory capacity when attempting to use large datasets. Whether these algorithms would be an improvement over others when

Table A.2. Training algorithm benchmark results. Average F-score and training time statistics are listed. These data are displayed in Figure A.1.

		Average F-score		Time (s)	
Algorithm	MATLAB function	Mean	St. dev.	Mean	St. dev.
Scaled conjugate gradient	`trainscg`	0.81	0.04	72	89
Polak–Ribiére Conjugate Gradient	`traincgp`	0.79	0.05	105	110
One Step Secant	`trainoss`	0.79	0.07	140	106
Gradient Descent (variable learning rate)	`traingdx`	0.79	0.06	160	29
Conjugate Gradient with Powell/Beale Restarts	`traincgb`	0.78	0.05	143	150
Fletcher–Powell Conjugate Gradient	`traincgf`	0.73	0.09	320	149
Resilient Backpropagation	`trainrp`	0.68	0.06	197	1

run on capable equipment remains a question.

Of the algorithms tested, the Scaled Conjugate Gradient (`trainscg`) produced networks providing the best and most consistent classification performance on the testing dataset, with a mean average F-score of 0.81. It also trained networks the fastest, with mean squared error falling below 2.5% after an average of 72 seconds.

As the Scaled Conjugate Gradient algorithm had the best training performance, both in time and in classification ability, I chose to use `trainscg` for all other neural network training in this project.

A.5 Neural Network Structure

The neural network structures used for the case studies in Chapter 3 (i.e. number of layers, number of neurons, number of delays, and other parameters) were selected after individual experimentation with each dataset. By some educated guessing as well as trial and error, we found the parameters that provided the best results for each problem. These specific trials will not be described here, but we do show two examples of how to determine an optimum number of layers and neurons.

First, we wanted to find out how many neurons are needed for an effective two-layer (i.e. one hidden layer and one output layer) network. We used the same training and testing data as the Katmai tests (Section 3.2), but ran several trials, varying the number of neurons in the hidden layer from 2 through 24 and attempting each ten times (for a total of 230 trials). We calculated the F-score for each resulting network, shown in summary in Figure A.2.

We can see from the figure that there was no significant improvement to F-scores for hidden layers with more than five neurons, and the best performance in this test was obtained with 10 neurons. This does not mean that 10 neurons are ideal for all earthquake classification problems, but it does provide a starting point for further tests.

The second test was to determine if performance improved by adding a second hidden layer. We used the same Katmai data as above, and varied the number of neurons in each layer from 2 through 8, running each test six times (for a total of 384 trials). We calculated the F-score for each resulting network, shown in Figure A.3.

We can see from the figure that adding a second hidden layer had no significant improvement to F-scores. The best scores were slightly over 0.8, which is comparable to the best two-layer networks. These three-layer networks took more than twice as long

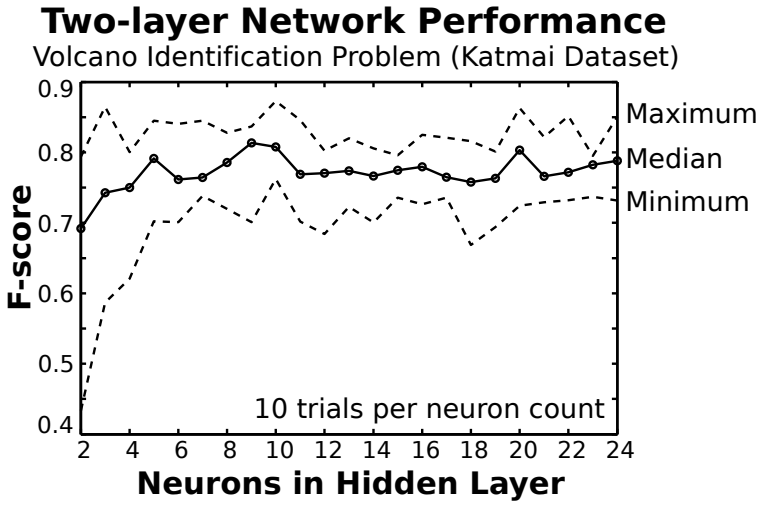

Figure A.2. Two-layer network performance. Median, minimum, and maximum F-score for varied numbers of hidden neurons in a two-layer network.

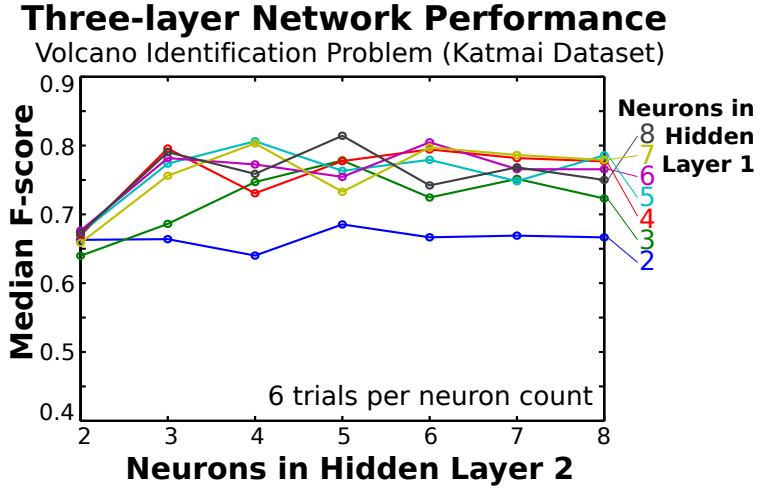

Figure A.3. Three-layer network performance. Median F-score for varied numbers of hidden neurons in two- and three-layer networks.

to train as the two-layer networks, with no significant performance improvement, thus we strongly recommend the use of only two-layer networks for this kind of classification problem.

A.6 Neuron Activation Function

The activation function, also known as the transfer function, is applied to the result of the weighted sum within each neuron in the network (Figure 2.1). A sigmoid function such as the logistic function (Equation A.17) or hyperbolic tangent function (Equation A.18) is commonly used in the hidden layers of the network, because it simplifies the output passing from one layer to the next.

$$\text{logistic}(x) = \frac{1}{1+e^{-x}} \tag{A.17}$$

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{A.18}$$

The exact shape of the sigmoid function is not particularly important, though it can affect the training speed [*Masters*, 1993]. We used the hyperbolic tangent function in all cases, as it was the default recommended for use with MATLAB's Neural Network Toolbox.

In some neural network applications, the final output layer uses a linear transfer function ($f(x) = x$). This is used if the output should be a real numerical result in a defined range, such as a magnitude or some other calculation result. In our case, we wanted output of either 0 or 1 (representing 'no' or 'yes' for a particular class), so a sigmoid function was most suitable in the output layer.

References

Beyreuther, M., and J. Wassermann, Continuous earthquake detection and classification using discrete Hidden Markov Models, *Geophysical Journal International*, *175*, 1055–1066, 2008.

Coombs, M. L., C. A. Neal, R. L. Wessels, and R. G. McGimsey, Geothermal disruption of summit glaciers at Mount Spurr Volcano, 2004–6: An unusual manifestation of volcanic unrest, in *Studies by the U.S. Geological Survey in Alaska, 2005: U.S. Geological Survey Professional Paper 1732-B*, edited by P. J. Haeussler and J. P. Galloway, United States Geological Survey, 2005.

De Angelis, S., and S. R. McNutt, Degassing and hydrothermal activity at Mt. Spurr, Alaska during the summer of 2004 inferred from the complex frequencies of long-period events, *Geophysical Research Letters*, *32*, 2005.

De Angelis, S., and S. R. McNutt, Observations of volcanic tremor during the January–February 2005 eruption of Mt. Veniaminof, Alaska, *Bulletin of Volcanology*, *69*, 927–940, 2007.

Falsaperla, S., S. Graziani, G. Nunnari, and S. Spampinato, Automatic classification of volcanic earthquakes by using multi-layered neural networks, *Natural Hazards*, *13*, 205–228, 1996.

Hinton, G., J. L. McClelland, and D. E. Rumelhart, Distributed representations, in *Parallel Distributed Representations: Explorations in the Microstructure of Cognition*, edited by E. Rumelhart and J. L. McClelland, vol. 1, pp. 77–109, MIT Press, Cambridge, MA, 1986.

Jay, J. A., M. E. Pritchard, M. E. West, D. Christensen, M. Haney, E. Minaya, M. Sunagua, S. R. McNutt, and M. Zabala, Shallow seismicity, triggered seismicity, and ambient noise tomography at the long-dormant Uturuncu Volcano, Bolivia, *Bulletin of Volcanology*, *74*, 817–837, 2012.

Köhler, A., M. Ohrnberger, and F. Scherbaum, Unsupervised pattern recognition in continuous seismic wavefield records using Self-Organizing Maps, *Geophysical Journal International*, *182*, 1619–1630, 2010.

Liu, C., P. Frazier, and L. Kumar, Comparative assessment of the measures of thematic classification accuracy, *Remote Sensing of Environment*, *107*, 606–616, 2007.

Masters, T., *Practical Neural Network Recipes in C++*, Academic Press, San Diego, CA, 1993.

Moran, S. C., S. D. Malone, A. I. Qamar, W. A. Thelen, A. K. Wright, and J. Caplan-Auerbach, Seismicity associated with renewed dome building at Mount St. Helens, 2004–2005, in *A Volcano Rekindled: The Renewed Eruption of Mount St. Helens, 2004–2006*, edited by D. R. Sherrod, W. E. Scott, and P. H. Stauffer, pp. 27–60, U.S. Geological Survey, 2008.

Møller, M. F., A scaled conjugate gradient algorithm for fast supervised learning, *Neural Networks*, *6*, 525–533, 1993.

Reyes, C. G., and M. E. West, The Waveform Suite: A robust platform for manipulating waveforms in MATLAB, *Seismological Research Letters*, *82*, 104–110, 2011.

Sokolova, M., and G. Lapalme, A systematic analysis of performance measures for classification tasks, *Information Processing and Management*, *45*, 427–437, 2009.

Sparks, R. S. J., C. B. Folkes, M. C. S. Humphreys, D. N. Barfod, J. Clavero, M. Sunagua, S. R. McNutt, and M. E. Pritchard, Uturuncu volcano, Bolivia: Volcanic unrest due to mid-crustal magma intrusion, *American Journal of Science*, *308*, 727–769, 2008.

Taner, M. T., F. Koehler, and R. E. Sheriff, Complex seismic trace analysis, *Geophysics*, *44*, 1041–1063, 1979.

Waibel, A., T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, Phoneme recognition using time-delay neural networks, *IEEE transactions on acoustics, speech, and signal processing*, *37*, 328–339, 1989.

Yıldırım, E., A. Gülbağ, G. Horasan, and E. Doğan, Discrimination of quarry blasts and earthquakes in the vicinity of Istanbul using soft computing techniques, *Computers & Geosciences*, *37*, 1209–1217, 2011.